NASA TM-83783

NASA Technical Memorandum 83783

NASA-TM-83783 19850002375

Floating-Point Function Generation Routines for 16-Bit Microcomputers

Michael A. Mackin and James F. Soeder Lewis Research Center Cleveland, Ohio

LIBRARY COPY

198 (198)

LANGLEY RESEARCH CENTER
LIBRARY, NASA
HAMPTON, VIRGINIA

October 1984



•
•

FLOATING-POINT FUNCTION GENERATION ROUTINES FOR 16-BIT MICROCOMPUTERS

Michael A. Mackin and James F. Soeder National Aeronautics and Space Administration Lewis Research Center Cleveland, Ohio 44135

SUMMARY

Several computer subroutines have been developed that interpolate three types of nonanalytic functions: univariate, bivariate, and map. The routines use data in floating-point form. However, because they are written for use on a 16-bit Intel 8086 system with an 8087 mathematical coprocessor, they execute as fast as routines using data in scaled integer form. Although all of the routines are written in assembly language, they have been implemented in a modular fashion so as to facilitate their use with high-level languages.

INTRODUCTION

The use of digital microcomputers in real-time simulation and control applications has created a need for high-speed function generation routines. These routines are needed to mathematically simulate the relationships between the inputs and outputs of a physical system.

For function generation routines being used in the control of systems such as gas turbine engines, speed of execution is a major concern. Previously, because of this speed requirement, function generation programs could only deal with data in an integer format (ref. 1). However, by using a mathematical floating-point hardware coprocessor, these routines can be extended to deal with data in floating-point form.

The programs described in this report are written in assembly language for use with the Intel 8086 microprocessor and the 8087 coprocessor chip. They use straight-line interpolation as the numerical estimation method and are callable from high-level languages. The ability of these programs to be directly linked with a high-level language makes it possible to develop the main logic of a control in a high-level language with calls to the assembly language routines when high-speed function generation is needed.

These programs deal with three types of functional relationships: (1) simple input-output functions (univariate functions), (2) functions of two variables with identical x data values on each curve (bivariate functions), and (3) functions of two variables with different x data values on each curve (map functions). Figures 1 to 3 illustrate each type. The programs are described by examining the three types of functions, by discussing the computer subroutines written to implement these functions, and by examining how the high computation speeds were achieved.

N85-10683#

DESCRIPTION OF FUNCTION TYPES

The six programs described in this report can interpolate outputs for three kinds of functional relationships. These functional relationships are described in this section.

Function Type One (Univariate Function)

This is the simplest type of function. It is a simple input-output relationship in which all data points are on the same curve (fig. 1). Every input value x uniquely identifies an output value y. The function can be symbolized as $Y_y = F(X_y)$. An output value Y_y for which there is no corresponding experimental data point X_y can be found by linearly interpolating between adjacent data points. This interpolation is carried out by using equation (1).

$$Y_{V} = \left(\frac{X_{V} - X_{L}}{X_{H} - X_{L}}\right) \left(Y_{H} - Y_{L}\right) + Y_{L} \tag{1}$$

The relationships between variables are shown in figure 1.

Function Type Two (Bivariate Function)

This function consists of a family of curves (fig. 2) instead of the single curve of function type one. Each curve has a particular value of z assigned to it. Therefore it is necessary to provide two input values, X_V and Z_V , before an output can be determined. This type of functional relationship can be symbolized as $Y_V = F(X_V, Z_V)$. For this function a restriction is made that each curve must have the same number of data points. Additionally, it is required that the data points for each curve be taken at identical values of x. The output Y_V is found by linearly interpolating between adjacent x and z values by using equations (2) to (4):

$$Y_{L} = \left(\frac{X_{V} - X_{L}}{X_{H} - X_{L}}\right) \left(Y_{B} - Y_{A}\right) + Y_{A}$$
 (2)

$$Y_{H} = \left(\frac{X_{V} - X_{L}}{X_{H} - X_{L}}\right) \left(Y_{D} - Y_{C}\right) + Y_{C}$$
 (3)

$$Y_{V} = \left(\frac{Z_{V} - Z_{L}}{Z_{H} - Z_{L}}\right) \left(Y_{H} - Y_{L}\right) + Y_{L}$$
(4)

The relationships between variables are shown in figure 4.

Function Type Three (Map Function)

This last function, which also consists of a family of curves, is the most general one. The restriction requiring all curves to have the same x breakpoints is removed. However, each curve is still required to have the same number of breakpoints per z curve (fig. 3). Because each curve has a unique set of x and y breakpoints, additional calculations must be made. The equations required to obtain the interpolated output $Y_V = F(X_V, Z_V)$ are

$$X_{F} = \begin{pmatrix} \frac{Z_{V} - Z_{L}}{Z_{H} - Z_{L}} \end{pmatrix} \begin{pmatrix} X_{B} - X_{C} \end{pmatrix} + X_{C}$$
 (5)

$$X_{G} = \begin{pmatrix} \frac{Z_{V} - Z_{L}}{Z_{H} - Z_{L}} \end{pmatrix} \begin{pmatrix} X_{E} - X_{D} \end{pmatrix} + X_{D}$$
 (6)

$$Y_{F} = \begin{pmatrix} \frac{Z_{V} - Z_{L}}{Z_{H} - Z_{L}} \end{pmatrix} \begin{pmatrix} Y_{B} - Y_{C} \end{pmatrix} + Y_{C}$$
 (7)

$$Y_{G} = \left(\frac{Z_{V} - Z_{L}}{Z_{H} - Z_{L}}\right) \left(Y_{E} - Y_{D}\right) + Y_{D}$$
 (8)

$$Y_{V} = \begin{pmatrix} x_{V} - x_{F} \\ x_{G} - x_{F} \end{pmatrix} \begin{pmatrix} Y_{G} - Y_{F} \end{pmatrix} + Y_{F}$$
 (9)

The relationships between variables are shown in figure 5.

DESCRIPTION OF COMPUTER ROUTINES

The programs used to implement function generation were developed for the Intel 8086 16-bit microprocessor coupled with an 8087 mathematical coprocessor. The 8086 chip uses 16-bit address registers and 16-bit segment registers. An address register and a segment register are combined to form a 20-bit address that can access up to 1 megabyte of storage (fig. 6). In Intel "small-memory model" programs the segment registers remain constant while the program is executing. Although this restriction allows the program to address only 64K bytes of storage, it simplifies many programming tasks and has the least amount of programming overhead. Intel "large-memory model" programs do not require the segment registers to remain constant. Thus large-model programs must set up and maintain the segment registers at their correct values. Because of the subtle differences between the large and small models, six separate function generation routines are presented in this report, a large- and small-model program for each of the three types of function. Further information on the large- and small-memory models for the 8086/8087 processors is given in references 2 to 5.

Linking to High-Level Languages

The function generation routines can be linked to both large- and small-model high-level language programs because all Intel conventions regarding the saving and restoring of registers are followed. This means that the routines can be used with Fortran-86, a large-model, high-level language, or PL/M-86, which can be either a large- or small-model language.

The high-level language must supply the function generation routines with two types of input parameter: (1) the address in memory of a data block holding information about the experimental breakpoints (the breakpoint information block), and (2) the input values of the function.

The breakpoint information block contains two types of information: (1) information used to describe the format of the experiment data, and (2) the arrays holding the data. All items in the breakpoint information block are identified in figure 7.

An example of a typical high-level-language call including setup of the breakpoint information block is shown in figure 8. According to parameter passing conventions, the first storage address in the breakpoint information block (the variable XPTR) is passed by the high-level-language program on the 8086 stack, and the real-number input values are passed on the 8087 stack.

The final interpolated result produced by the lookup routine is returned to the high-level-language program on the 8087 stack. When the parameters are passed to the subroutine as shown in figure 8, this process is taken care of automatically by the compiler and assembly language function routines and is transparent to the high-level language programmer.

Out-Of-Range Conditions

For a typical control application an out-of-range data input must not be allowed to cause an unpredictable output. This is necessary to prevent damage to the control system that could occur if, for instance, an actuator were pushed beyond its range of operation. If a value is out or range, either too high or too low, the programs will output the closest boundary value of the function. If, for example, the input value is higher than the last experimental data point in the breakpoint information block of a simple input-output curve, the routine will output the highest experimental value. If the input z value into a two-variable function interpolation routine is above the maximum z data value, the routine will use the highest allowable z curve when performing the interpolation (fig. 9).

Dealing with out-of-range values in this way keeps the output from becoming unpredictable and results in a control system that is well behaved.

OPERATION OF COMPUTER ROUTINES

Six subroutines are described in this report, two for each type of function: They are RFUNIS, RFUNIL, RFUN2S, RFUN2L, RFUN3S, and RFUN3L. The last letter in the program name tells the type of memory model with which the

program will interface. A "S" suffix indicates that the routine is to be used in the small-memory model, and a "L" indicates use with the large-memory model. Listings of all programs, including example calls, are included in the appendix to this report.

RFUNIS and RFUNIL Programs

The routines RFUNIS and RFUNIL, which interfaces with the Intel small—and large-model programs respectively, deal with simple univariate input-output function lookups. The data are set up in the breakpoint information block as shown in figure 10. Figure 8 shows how these block data are set up by a Fortran calling program.

XPTR is an index pointing to the location of the data in the most recent previous call to the routine. ZPTR is not used in these routines but is included to make the data structure similar to that used in RFUN2 and RFUN3. NXPTS is the number of data breakpoints, NZPTS is not used but is included for the same reason as ZPTR. XARRAY, which contains the breakpoint values of x, is stored next followed by YARRAY, the function output for each x value.

During execution three basic operations are performed. First the routine initializes its registers. Then it determines whether the input value is higher or lower than the initial x value indicated by the XPTR index. The program then scans the XARRAY table to find the x segment that corresponds to the x input value. Once the scanning is complete, the 8087 chip is used to perform the interpolation by using equation (1), which was discussed earlier. Finally, the routine updates the index pointer and returns to the calling program with the interpolated function output value y on the 8087 stack.

RFUN2S and RFUN2L Programs

RFUN2S and RFUN2L generate values for two-variable functions with identical breakpoints (as in fig. 2) in the small- and large-model programs, respectively. They function similarly to RFUNIS and RFUNIL with one additional step. The routines must perform a search through the z values as well as through the x values. They must also read and update the ZPTR value in the breakpoint information block (this variable was not used in the RFUNI programs).

The data are stored in this program as shown in figure 11. Since all of the x breakpoints occur at identical points on all curves, only a one-dimensional XARRAY is necessary. However, because each curve will have a different y output value at each breakpoint, a two-dimensional YARRAY is needed. The arrangement of this array is such that the first row corresponds to the outputs of the first z curve, the second row corresponds to the outputs of the second z curve, and so on, up to the number of curves that exist.

The programs make use of equations (2) to (4) to compute the proper interpolated output. As in the RFUN1 programs, the final interpolated function output is returned on the 8087 stack.

RFUN3S and RFUN3L Programs

The RFUN3 programs generate outputs for two-variable functions with individual breakpoints (as in fig. 3). They operate similarly to RFUN2S and RFUN2L except that additional calculations are made during each x scan to find the breakpoints on a new z curve. The RFUN3 programs first scan the ZARRAY to determine which curves the z input value is between. Next it uses equations (5) and (6) to generate the x breakpoints of this new interpolated z curve. The x value search proceeds by generating new x values as a function of z until the upper and lower boundaries of the input x are found. Then interpolation between the generated segment endpoints by using equations (7) to (9) determines the final output value y. Finally the x and z index pointers are updated and the y output value is returned to the calling program on the 8087 stack.

The data setup for the RFUN3 programs is more complicated than that for either the RFUN1 programs or the RFUN2 programs because each breakpoint is independent of all others. Not only does each curve have an individual set of y output values, as in the RFUN2 programs, but each also has an individual set of x values. To support this arrangement, the data are set up as shown in figure 12. XARRAY is a two-dimensional array containing the breakpoints for each curve. The first row corresponds to the first z curve, the second row to the second z curve, and so on as in the YARRAY used in the RFUN2 programs. The YARRAY used in the RFUN3 programs is set up the same as the YARRAY in the RFUN2 programs.

IMPLEMENTING FAST EXECUTION SPEEDS

Since function generation routines are used most often in real-time applications where execution time is critical, an efficient algorithm is very important. Several techniques are used in the function generation routines to attain this speed efficiency.

Before interpolation is possible, the function generation routine must determine the magnitude of the input by searching through the breakpoint information block values for the one closest to the input value. Typically, since a specific function routine is called only once per control update interval, and since the input value normally changes relatively slowly from one control interval to the next, the breakpoint information block search is sped up if the index location of the previous search is known. All of the function generation programs described in this report store the final index value for use as the starting point of the next search. This index value, XPTR for example, is stored at the beginning of the breakpoint information block and is updated at the end of each call of the function generation routine.

Another way to decrease the execution time of the routine is to use memory access instructions only when necessary since these take longer to execute than register access instructions. The routines described in this report store any frequently used integer value in an 8086 internal register and any frequently used real value in an 8087 register, thus eliminating unnecessary memory accesses.

A third approach used to decrease the execution time is to use in-line coding of instructions rather than subroutine calls. In-line coding may cause the size of the code to be larger, but it will execute faster because the processor does not have to store pointers and status registers as it does for subroutine calls.

Function Routine Execution Times

Typical execution times for the entire lookup routine were experimentally determined to be 0.50 ms for a univariate function lookup, 0.90 ms for a bivariate function lookup, and 1.00 ms for a map function lookup.

CONCLUDING REMARKS

Three types of nonanalytic function routines have been developed for use with control algorithms written in high-level languages. These routines operate on data in floating-point form and are fast enough to be used in real-time control systems. Additionally, problems associated with the 8086 segmented architecture have been resolved so that the routines can be easily and efficiently integrated into most control schemes.

REFERENCES

- Soeder, James F.; and Shaufl, Maryrita: Nonanalytic Function Generation Routines for 16-Bit Microprocessors. NASA TM-81586, 1980.
- 2. MCS-86 Macro Assembly Language Reference Manual. Intel Corp. (Manual Order No. 9800640-02), 1979.
- 3. MCS-86 Assembler Operating Instructions for ISIS-II Users. Intel Corp. (Manual Order No. 9800641A), 1978.
- 4. PL/M-86 Users Guide. Intel Corp. (Order No. 121636-003), 1982.
- 5. FORTRAN-86 Users Guide. Intel Corp. (Order No. 121570-001), 1981.

APPENDIX - PROGRAM LISTINGS

SERIES-III 8086/87/83/186 MACRO ASSEMBLER V2.0 ASSEMBLY OF MODULE RFUNIS OBJECT MODULE PLACED IN :F1:RFUN1S.OBJ ASSEMBLER INVOKED BY: ASM86.86 :F1:RFUNIS.SRC

LOC OBJ	LINE	SOURCE					
	1	;	************************				
	2	•	* ROUTINE NAME: REUNIS *				
	3	;	* VERSION: 2.0 *				
	4	;	* DATE: AUG. 9, 1983 *				
	5	•	* PROGRAMMER'S NAME: MICHAEL MACKIN *				
	6	;	* PURPOSE: TO INTERPOLATE A SIMPLE *				
	7	;	* FUNCTION FOR SMALL-MODEL *				
	8	;	* PROGRAMS *				
	9	;	* ************************************				
	10	;	00000000000000000000000000000000000000				
	11	:					
	12	;	THIS SUBROUTINE OUTPUTS THE ESTIMATED VALUE OF A FUNCTION $YV = F(XV)$				
	13	;	BY INTERPOLATING FROM KNOWN SOLUTION POINTS, USING THE RELATION				
	14	•	YV = ((XV-XL)/(XH_XL))(YL-YL) + YL WHERE				
	15	;	YV = ANSWER TO BE COMPUTED				
	16	;	XV = INPUT VALUE				
	17	;	XH = KNOWN HIGH X FOINT				
	18	;	XL = KNOWN LOW X POINT				
	19	;	YH = KNOWN HIGH Y POINT				
	20	;	YL = KNOWN LOW Y POINT				
	21		IC - MICHAEL FOR 1 1 OTHE				
	22	;	INPUT REQUIREMENTS:				
	23	•	1) THE FOLLOWING ADDRESSES WILL BE PASSED TO THIS SUBROUTINE:				
	24	;	A. ON TOP OF THE 8086 STACK:				
	25	;	THE ADDRESS (OFFSET) OF A DATA AREA WITH THE FOLLOWING				
	26	;	FORMAT				
	27	;	: Ottain				
	23		DSTRUC STRUC				
	29		411164				
0000	30		XPTR DW ? ;X ARRAY INDEX				
0002	31		ZPTR DW ? ;Y ARRAY INDEX				
0004	32		NXPTS DW ? ;NO. OF ELEMENTS IN X ARRAY				
0006	33		NZPTS DW ? ;NO. OF ELEMENTS IN Y ARRAY				
	34	;	XARRAY DD NXPTS DUP(?) ;X ARRAY				
	35	;	YARRAY DD NYPTS DUP(?) ;Y ARRAY				
	36						
	37		DSTRUC ENDS				
	38	í					
	39	;	B. ON TOP OF THE 8087 STACK:				
	40	;	THE INPUT VALUE, XV, TO BE INTERPOLATED				
	41	;					
	42	;	2) IT IS REQUIRED THAT THE 8087 CHIP HAVE 3 EMPTY REGISTERS				
	43	;	WHEN THIS INTERPOLATION ROUTINE IS CALLED.				
	44	;					
	45	;	OUTPUT EFFECTS:				
	46	;	1) REGISTERS DESTROYED: AX, BX, CX, DX, DI, SI				
	47	f	INTERPOLATED RESULT RETURNED ON TOP OF 8087 STACK				
	48	;	 XPTR LOCATION UPDATED TO INDEX X VALUE JUST PRECEDING XV. 				
	49	;	(TO SPEED FUTURE SEARCHES FOR X)				
	50	;					

2

```
LOC OBJ
                    LINE
                           SOURCE
                                 EXAMPLE PL/M PROGRAM USING RFUNIS:
                      51
                           ţ
                      52
                           ;
                                 $SMALL
                      53
                           ;
                      54
                      55
                                 PLMPGH: DO:
                      56
                      57
                                        DECLARE DSTRUC STRUCTURE (
                                          XINDEX INTEGER, ZINDEX INTEGER, NXPTS INTEGER, NZPTS INTEGER,
                      58
                                          XARRAY(7) REAL, YARRAY(7) REAL) PUBLIC
                      59
                                          INITIAL (0,0,7,7,0,0,1.0,2.0,3.0,4.0,5.0,6.0,0.0,1.0,
                      60
                                                 4,0,9,0,16,0,25,0,36,0);
                      61
                           ;
                      62
                                        RFUNIS: PROCEDURE(X,STRUC_ADDR) REAL EXTERNAL;
                      63
                                              DECLARE X REAL;
                      64
                                              DECLARE STRUC_ADDR POINTER;
                      65
                           ï
                      66
                           ;
                                        END RFUNIS;
                      67
                           ;
                                        PLMTEST: PROCEDURE PUBLIC;
                      68
                           ;
                                              DECLARE I INTEGER;
                      69
                                              DECLARE (X,Y) REAL;
                      - 70
                      71
                                                X = 1.5
                                                 Y = RFUNIS(X, @DSTRUC);
                      72
                                                 CALL PRINT_ANSWER(Y);
                      73
                           ï
                      74
                                              END;
                      75
                                        END FUNPOM:
                      76
                      77
                           ;
                                  END PLMPGM;
                      73
                            79
                      80
                      81
                                        RFUNIS
                      82
                                  NAME
                                       CODE
                      83
                            CGROUP GROUP
                      84
                            DGROUP GROUP DATA
                                  ASSUME CS:CGROUP, DS:DGROUP
                      85
                                  PUBLIC RFUNIS
                      86
                      87
                            <del></del>
                      88
                      89
                                  STACK STRUCTURE
                      90
                      91
                            PARAMS STRUC
                      92
                      93
                                        ?
                            OLD_BP DW
                                               SAVED BP REGISTER
0000
                      94
                                         ?
                      95
                            RETURN DW
                                               RETURN ADDRESS
0002
                                               ADDRESS OF DATA STRUCTURE
                      96
                            DADDR DW
0004
                      97
                       98
                            PARAMS ENDS
                      99
                            100
                            101
                      102
                                  SEGMENT PUBLIC 'DATA'
                      103
                            DATA
                      104
                                         ?
                      105
                            STWSAVE DW
 0000 ????
```

LOC OBJ		LINE	SOURCE					
		106 107 108	DATA	ENDS				
		109 110						*******************************
		111 112 113	COD€	SEGMENT	PUBLI	C /CODE/		
0000		114 115 116	NEARCOD	E	PROC	NEAR		
0000		117 118	RFUN1S:					
0000 55		119 120		PUSH	₽P		;SAVE	REGISTERS
0001 8BEC 0003 8B5E04 0006 8B3F		121 122 123		VOM VOM VOM	BP, SI BX, CI DI, CI	5 6PJ.DADDR 8XJ.XPTR		START ADDRESS OF DATA STRUCTURE ELEMENT INDEX
0008 D1E7 000A D1E7 000C 8B7704 000F D1E6		124 125 126 127		SHL SHL MOV SHL	DI, 1 DI, 1 SI, []	BX].NXPTS	;DI = ;SI =	BYTE INDEX = ELEMENT INDEX * 4 NXPTS
0011 D1E6 0013 83C308 0016 8BD3		128 129 130		SHL ADD MOV	SI, 1 BX, 8 DX, B		;BX =	NO. OF BYTES=NXPTS * 4 START OF X ARRAY
0018 03D6 001A 9BD811 001D 9BDD3E0000 0022 9B	R	131 132 +2 133 +1 134 +1		ADD FCOM FSTSW FWAIT	DX, SI DWORD STUSA	PTR [BX + DI]	;DX =	START OF Y ARRAY COMPARE ARRAY ELEMENT WITH XV
0023 A10000 0026 250041 0029 3D0001	R	135 +1 136 +1 137 +2		MOV AND CMP	AX, S' AX, 4: AX, 0:		;	MASK-IN COMPARISON BITS
002C 7432		133 +2 139 +2 140 +1		JE		DF_START		: EXIT IF XV < ARRAY ELEMENT
002E		141 142	RIGHT_0	F_START:				> DATA(ORIGINAL INDEX) THEN BEGIN ***
002E 83C704		143		ADD	DI, 4		i i	IDEX = INDEX + 4
0031 3BFE		144	RS1:	CMP	DI, SI	Ī	; (#	MILE (INDEX <= NO. OF BYTES)
0033 741E		145 146		JE	LI		7	*** DO PEGIN ***
0035 9BD311		147 +2		FCOM			;	COMPARE ARRAY ELEMENT WITH XV
0038 9BDD3E0000 003D 9B	R	148 +1 149 +1		FSTSW FWAIT	STWSAV	Æ		
003E A10000	R	150 +1		HOV	AX, ST	WSAVE		
0041 250041		151 +1		AND	AX, 41		;	MASK-IN COMPARISON BITS
0044 3D0001 0047 740A		152 +2		CMP	AX, 01	H00H		
0047 740A 0049 3D0040		153 +2 154 +2		JE CMP	L1 AX: 40	ากกน	;	EXIT IF XV < ARRAY ELEMENT
004C 7405		155 +2 156 +1		JE JE	Li	700H	;	EXIT IF XV = ARRAY ELEMENT
604E 83C704		157		ADD	DI, 4		;	INDEX = INDEX + 4
0051 EBDE		158		JMP	RS1		;	*** END (WHILE) ***
0053 83EE04		159	L1:	SUB	SI, 4		;	
0056 83EF04		160		SUB	DI, 4		;	

Compared 161	LOC	OBJ		LINE	SOURCE			
163	0059	3BFE		161		CMP	DI, SI	; IF INDEX > NO. OF BYTES
164 185	0058	7458		162		JE	OUT_OF_RANGE	; THEN OUT_OF_RANGE
155 156 157 158	005B	EB3290		163		JMP	FOUND_LOW_INDEX	; ELSE FOUND_LOW_INDEX
166				164				;*** END (IF) ***
0060 8SEF04				165				; IF X < DATA(ORIGINAL INDEX)
Mark	0060			166	LEFT_OF.	_START:		;*** THEN BEGIN ***
OSS 38FF00							DI, 4	; INDEX = INDEX - 4
100					131:			
170					C01.			•
ONE PRODUCTION R	VV30	/CIE				OL.	112	* *** DO REGIN ***
0068 98D0500000 R 172 +1 FSTSW STNSAVE 0070 98 173 +1 FMAIT 0071 A10000 R 174 +1 HOV AX. STNSAVE 0074 250041 175 +1 A80 AX. 4100H : MASK-IN COMPARISON BITS 0077 3D0000 176 +2 CMP AX. 0000H : EXIT IF XV > ARRAY ELEMENT 0077 3D0000 177 +2 JE H1 0070 3B0040 173 +2 JE H1 : EXIT IF XV > ARRAY ELEMENT 0070 7B0040 173 +2 JE H1 : EXIT IF XV > ARRAY ELEMENT 0070 7B0040 180 +1 0081 3B5F04 181 SUB DI, 4 ; INDEX = INDEX - 4 0084 EBD0 182 MP LS1 ; *** EMD (MATLE) *** 0088 B3F700 183 H1: CMP DI, 0 ; IF INDEX ⊃ INDEX.MIN 0088 B3C704 185 ABD DI, 4 ; 0088 B3C704 185 ABD DI, 4 ; 0088 B3C704 185 ABD DI, 4 ; 0091 189 FOUND_LOW_INDEX: 0091 189 FOUND_LOW_INDEX: 0091 189 FOUND_LOW_INDEX: 0091 190 HTERFOLATE: 0091 190 STRONG PTR (EX+DI) 0091 190 STRONG PTR (EX+DI) 0097 76004904 193 FSUER LWARD PTR (EX+DI) 0097 76004804 193 FSUER LWARD PTR (EX+DI) 0097 76004904 195 XCHG BX, DX 0009 9B001 196 FID DARRO PTR (EX+DI) 0098 9B001 196 FID DARRO PTR (EX+DI) 0098 9B001 197 FID DARRO PTR (EX+DI) 0098 9B001 198 FSUB ST, ST(1), ST (XM-XL) ON STACK TOP 0099 9B001 199 FWLP 0009 9B001 199 FWLP	0010	000011		-		ECOM	דות ב עמן מדם המחוות	
0070 BB			ь					, CONTRAC MARKET CECHERI WITH AV
OPT ALGOROUM 175 + 1			К				PIMPHAE	
175 + 1							AV OTHORUE	
0077 3D0000 176 +2 CMP AK, 0000H ; EXIT IF XV > ARRAY ELEMENT 0077 740A 177 +2 JE M1			н					. MACK IN COMPARISON DITC
007A 7408 177 +2								
OPTC 300040								; EXII IF XV > ARRAY ELEMENT
199								
190 +1 191 +1	007C	3D0040		178 +2		CMP		
ONE 838F04 181	007F	7405		179 +2		JΕ	Mi	; EXIT IF XV = ARRAY ELEMENT
192				180 +1				
0086 83FF00 183 H1: CMP DI. O ; IF INDEX ◇ INDEX_MIN 0087 7D06 184	0081	83EF04		181		SUB	DI, 4	; INDEX = INDEX - 4
COS9 7006 194	0084	EBDD		182		JMP	LS1	; *** END (WHILE) ***
COS9 7D06	9880	83FF00		183	H1:	CMP	DI, O	; IF INDEX ⇔ INDEX_MIN
NOBB 83704 185				184		JGE	FOUND_LOW_INDEX	; THEN FOUND_LOW_INDEX
186								n 1
187								FLSE OUT_OF_RANGE:
188		LDLOVO				0.11	221201210112	
0091								y and the half of the second
190	0091				FOUND I	OU INDEX	(:	
0.91 9BD901 191 FLD DNORD PTR (BX+DI1 STORE XL ON STACK TOP 0.094 9BDCE9 192 FSUB ST(1), ST ;(XV-XL) ON STACK BOTTOM 0.097 9BDS4904 193 FSUER DNORD PTR (BX+DI+4) ;(XH-XL) ON STACK BOTTOM 0.098 9BDEF9 194 FDIV ;(XV-XL)/(XH-XL) LEFT ON STACK 0.098 87DA 195 XCH6 BX, DX 0.008 9BD901 196 FLD DNORD PTR (BX+DI+4) ;STORE YL ON STACK TOP 0.008 9BD901 197 FLD DNORD PTR (BX+DI+4) ;STORE YL ON STACK TOP 0.008 9BD91 198 FSUB ST, ST(1) ;(YH-YL) ON STACK TOP 0.008 9BDEC1 198 FSUB ST, ST(1) ;(YH-YL) ON STACK TOP 0.008 9BDEC1 200 FADD ;YV ON STACK TOP (ANSWER) 0.008 9BDEC1 200 FADD ;YV ON STACK TOP (ANSWER) 0.008 9BD90 202 JMP SAVE_NEN_XPTR 0.008 9BD90 205 FST ST(0) ;DISCARD TOP OF STACK 0.008 9BD90 205 FST ST(0) ;DISCARD TOP OF STACK 0.008 9BD90 207 FLD DNORD PTR (BX + DI) ;SAVE BOUNDRY Y VALUE 0.008 9BD90 209 XCH6 DX, BX 0.008 9BD90 209 SAVE_NEN_XPTR; 0.008 9BD90 209 SAVE_NEN_XPTR; 0.008 9BD90 209 SAVE_NEN_XPTR; 0.009 9BDEF 211 SHR DI, 1 0.001 DIEF 211 SHR DI, 1 0.002 83EB08 213 SUB BX, 8 1BX = XPTR LOCATION 0.006 893F 214 MOV WORD PTR (BX), DI ;SAVE NEN XPTR							\ •	YU ALREADY ON THE BY PLM POM
O94 9BDCE9 192					INIEM C		ווחבעם מדם ומעבות	
0097 98D86904 193 FSUBR DWORD PTR (BX+DI+4] :(XH-XL) ON STACK TOP 0098 9BDEF9 194 FDIV ;(XV-XL)/(XH-XL) LEFT ON STACK 0098 87DA 195 XCHG BX, DX 0000 98D901 196 FLD DWORD PTR (BX+DI) :STORE YL ON STACK TOP 0000 98D901 197 FLD DWORD PTR (BX+DI+4] :STORE YL ON STACK TOP 0000 98D81 198 FSUB ST, ST(1) :(YH-YL) ON STACK TOP 0000 98D8C1 199 FMULP ST(2), ST :(YH-YL) ON STACK TOP 0000 98D8C1 200 FADD :YV ON STACK TOP(ANSWER) 0000 98D8C1 201 XCHG BX, DX 0000 87DA 201 XCHG BX, DX 0000 87DA 201 XCHG BX, DX 0000 98DD00 202 JMP SAVE_NEN_XPTR 0000 98DD00 205 FST ST(0) :DISCARD TOP OF STACK 0000 87D3 204 XCHG DX, BX 0000 87D3 208 XCHG DX, BX 0000 87D								
009E 9BDEF9								
005E 87BA 195 XCHG BX, DX 00A0 9BD901 196 FLD DWORD PTR [BX+DI3] ;STORE YL ON STACK TOP 00A3 9BD94104 197 FLD DWORD PTR [BX+DI+4] ;STORE YH ON STACK TOP 00A7 9BD8E1 198 FSUB ST, ST(1) ;(YH-YL) ON STACK TOP 00AA 9BD8CA 199 FMULP ST(2), ST ;((XV-XL)/(XH-XL))(YH-YL)ON STACK BOT 00AD 9BD8C1 200 FADD ;YV ON STACK TOP (ANSWER) 00B0 87DA 201 XCHG BX, DX 00B2 EB0B90 202 JMP SAVE_NEW_XPTR 00B5 204 OUT_OF_RANGE: *** 00B5 9BDD00 205 FST ST(0) ;BISCARD TOP OF STACK 00B0 87D3 206 XCHG DX, BX *** 00B0 87D3 208 XCHG DX, BX *** 00B1 9BD901 207 FLD DWORD PTR [BX + DI] ;SAVE BOUNDRY Y VALUE 00B0 87D3 208 XCHG DX, BX *** 00B1 9BD901 207							EMOND FIR LEATELT 43	
00A0 9BD901 196 FLD DWORD PTR [BX+DI] :STORE YL ON STACK TOP 00A3 9BD94104 197 FLD DWORD PTR [BX+DI+4] :STORE YH ON STACK TOP 00A7 9BD8E1 198 FSUB ST, ST(1) :(YH-YL) ON STACK TOP 00AA 9BDECA 199 FMULP ST(2), ST :((XV-XL)/(XH-XL))(YH-YL)ON STACK BOT 00AD 9BDEC1 200 FADD :YV ON STACK TOP (ANSWER) 00B0 87DA 201 XCHG BX, DX 00B1 202 JMP SAVE_NEW_XPTR 00B2 EB0B90 203 OUT_OF_RANGE: 00B5 204 OUT_OF_RANGE: :DISCARD TOP OF STACK 00B8 87D3 206 XCHG DX, BX 00B8 9BD901 207 FLD DWORD PTR [BX + DI] :SAVE BOUNDRY Y VALUE 00BB 87D3 208 XCHG DX, BX :SAVE BOUNDRY Y VALUE 00BF D1EF 210 SAVE_NEW_XPTR: :DI = DI / 4 00BF D1EF 211 SHR DI, 1 :DI = DI / 4 00C3 83EB08 213 <							מע. מע	TAV ACTIVAL ACT COLOR OTHER
00A3 9BD94104 197 FLD DWORD PTR (BX+DI+4] STORE YH ON STACK TOP 00A7 9BD8E1 198 FSUB ST, ST(1) :(YH-YL) ON STACK TOP 00AA 9BDECA 199 FMULP ST(2), ST :((XV-XL)/(XH-XL))(YH-YL)ON STACK BOT 00AD 9BDEC1 200 FADD ;YV ON STACK TOP(ANSWER) 00B0 87DA 201 XCHG BX, DX 00B2 EB0B90 202 JMP SAVE_NEW_XPTR 203 00B5 204 OUT_OF_RANGE: 00B5 9BDDD0 205 FST ST(0) :DISCARD TOP OF STACK 00B3 87D3 206 XCHG DX, BX 00BA 9BD901 207 FLD DWORD PTR (BX + DI) :SAVE BOUNDRY Y VALUE 00BD 87D3 208 XCHG DX, BX 00BF 210 SAVE_NEW_XPTR: 00BF D1EF 211 SHR DI, 1 :DI = DI / 4 00C1 D1EF 212 SHR DI, 1 00C3 83EB08 213 SUB BX, 8 :BX = XPTR LOCATION 00C6 893F 214 MOV WORD PTR (BX), DI :SAVE NEW XPTR								• CTODE VI ON CTACK TOD
00A7 9B08E1 198 FSUB ST, ST(1) :(YH-YL) ON STACK TOP 00AA 9B0ECA 199 FMULP ST(2), ST :((XV-XL)/(XH-XL))(YH-YL)ON STACK BOT 00AD 9B0EC1 200 FADD ;YV GN STACK TOP(ANSWER) 00B0 87DA 201 XCHG BX, DX 00B2 EB0B90 202 JMP SAVE_NEW_XPTR 00B5 204 OUT_OF_RANGE: 00B5 9B0D0 205 FST ST(0) ;DISCARD TOP OF STACK 00B3 87D3 206 XCHG DX, BX 00B0 87D3 208 XCHG DX, BX 00BF 210 SAVE_NEW_XPTR: 00BF D1EF 211 SHR DI, 1 ;DI = DI / 4 00C1 D1EF 212 SHR DI, 1 ;DI = DI / 4 00C3 83EB08 213 SUB BX, 8 ;BX = XPTR LOCATION 00C6 893F 214 MOV WORD PTR [BX], DI ;SAVE NEW XPTR								
00AA 9BDECA 199 FMULP ST(2), ST ;((XV-XL)/(XH-XL))(YH-YL)ON STACK BOT OOAD 9BDEC1 200 FADD ;YV OR STACK TOP(ANSWER) 00B0 87DA 201 XCHG BX, DX BX, DX 00B2 EB0B90 202 JMP SAVE_NEW_XPTR SAVE_NEW_XPTR 00B5 9BDD00 205 FST ST(0) ;DISCARD TOP OF STACK 00B5 9BD901 206 XCHG DX, BX 00B0 9BD901 207 FLD DWORD PTR (BX + DI) ;SAVE BOUNDRY Y VALUE 00B0 87D3 208 XCHG DX, BX 00BF D1EF 210 SAVE_NEW_XPTR: 00BF D1EF 211 SHR DI, 1 00C1 D1EF 212 SHR DI, 1 00C3 83EB08 213 SUB BX, 8 00C6 893F 214 MOV WORD PTR (BX), DI ;SAVE NEW XPTR								
00AB 9BDEC1 200 FADD ; YV ON STACK TOP (ANSWER) 00B0 87DA 201 XCHG BX, DX 00B2 EB0B90 202 JMP SAVE_NEW_XPTR 203 204 OUT_OF_RANGE: **DISCARD TOP OF STACK** 00B5 9BDD0 205 FST ST(0) **IDISCARD TOP OF STACK** 00B8 87D3 206 XCHG DX, BX **SAVE BOUNDRY Y VALUE** 00B0 87D3 208 XCHG DX, BX **SAVE BOUNDRY Y VALUE** 00BF D1EF 210 SAVE_NEW_XPTR** **DI = DI / 4 00C1 D1EF 211 SHR DI, 1 **IDI = DI / 4 00C3 83EB08 213 SUB BX, 8 **BX = XPTR LOCATION** 00C6 893F 214 MOV WORD PTR [BX], DI **SAVE NEW XPTR**								
00B0 37DA 201 XCHG BX, DX 00B2 EB0B90 202 JMP SAVE_NEW_XPTR 203 204 OUT_OF_RANGE: 203 00B5 9BDDD0 205 FST ST(0) ;DISCARD TOP OF STACK 00B8 87D3 206 XCHG DX, BX 00BA 9BD901 207 FLD DWORD PTR [BX + DI] ;SAVE BOUNDRY Y VALUE 00BD 87D3 208 XCHG DX, BX 00BF 210 SAVE_NEW_XPTR: 00BF 210 SAVE_NEW_XPTR: 00BF D1EF 211 SHR DI, 1 ;DI = DI / 4 00C1 D1EF 212 SHR DI, 1 ;DI = DI / 4 00C3 83EB08 213 SUB BX, 8 ;BX = XPTR LOCATION 00C6 893F 214 MOV WORD PTR [BX], DI ;SAVE NEW XPTR							511/2/1 51	
0082 EB0890 202 JMP SAVE_NEN_XPTR 203 203 00B5 204 OUT_OF_RANGE: 0085 9BDDD0 205 FST ST(0) ;DISCARD TOP OF STACK 0088 87D3 206 XCHG DX, BX 00BA 9BD901 207 FLB DWORD PTR [BX + DI] ;SAVE BOUNDRY Y VALUE 00BD 87D3 208 XCHG DX, BX 00BF 210 SAVE_NEW_XPTR: SAVE_NEW_XPTR: 00BF D1EF 211 SHR DI, 1 ;DI = DI / 4 00C1 D1EF 212 SHR DI, 1 ;DI = DI / 4 00C3 83EB08 213 SUB BX, 8 ;BX = XPTR LOCATION 00C6 893F 214 MOV WORD PTR [BX], DI ;SAVE NEW XPTR							5V 5V	TYV UN STACK TUP (ANSWER)
203 204 OUT_OF_RANGE:								
00B5 204 OUT_OF_RANGE: OUBS 9BDDDO 205 FST ST(0) ;DISCARD TOP OF STACK 00B8 87D3 206 XCHG DX, BX DX, BX OUBS 9BD901 207 FLD DWORD PTR [BX + DI] ;SAVE BOUNDRY Y VALUE DX, BX OUBS 9BD901 SAVE BOUNDRY Y VALUE OUBS 9BD901 OUBS 9BD902	0082	: EB0B90				JMP	SAVE_NEW_XPTR	
00B5 9BDD00 205 FST ST(0) ;DISCARD TOP OF STACK 00B8 87D3 206 XCHG DX, BX 00BA 9BD901 207 FLD DWORD PTR [BX + DI] ;SAVE BOUNDRY Y VALUE 00BB 87D3 208 XCHG DX, BX 209 209 DEF 210 SAVE_NEW_XPTR: 00BF D1EF 211 SHR DI, 1 ;DI = DI / 4 00C1 D1EF 212 SHR DI, 1 ;DI = DI / 4 00C3 83EB08 213 SUB BX, 8 ;BX = XPTR LOCATION 00C6 893F 214 MOV WORD PTR [BX], DI ;SAVE NEW XPTR								
00B8 87D3					OUT_OF.			
00BA 9BD901 207 FLD DWORD PTR [BX + DI] ;SAVE BOUNDRY Y VALUE 00BD 87D3 208 XCHG DX, BX 209 00BF 210 SAVE_NEW_XPTR: 00BF D1EF 211 SHR DI, 1 ;DI = DI / 4 00C1 D1EF 212 SHR DI, 1 00C3 83EB08 213 SUB BX, 8 ;BX = XPTR LOCATION 00C6 893F 214 MOV WORD PTR [BX], DI ;SAVE NEW XPTR								DISCARD TOP OF STACK
00BD 87D3 208 XCHG DX, BX 209 00BF 210 SAVE_NEW_XPTR: 00BF D1EF 211 SHR DI, 1 ;DI = DI / 4 00C1 D1EF 212 SHR DI, 1 00C3 83EB08 213 SUB BX, 8 ;BX = XPTR LOCATION 00C6 893F 214 MOV WORD PTR [BX], DI ;SAVE NEW XPTR								
209 00BF 210 SAVE_NEW_XPTR: 00BF D1EF 211 SHR DI, 1 ;DI = DI / 4 00C1 D1EF 212 SHR DI, 1 00C3 83EB08 213 SUB BX, 8 ;BX = XPTR LOCATION 00C6 893F 214 MOV WORD PTR [BX], DI ;SAVE NEW XPTR	00BA	9BD901						ISAVE BOUNDRY Y VALUE
00BF 210 SAVE_NEW_XPTR: 00BF D1EF 211 SHR DI, 1 ;DI = DI / 4 00C1 D1EF 212 SHR DI, 1 00C3 83EB08 213 SUB BX, 8 ;BX = XPTR LOCATION 00C6 893F 214 MOV WORD PTR [BX], DI ;SAVE NEW XPTR	OOBI) 87D3				XCHG	DX, BX	
00BF D1EF 211 SHR DI, 1 ;DI = DI / 4 00C1 D1EF 212 SHR DI, 1 00C3 83EB08 213 SUB BX, 8 ;BX = XPTR LOCATION . 00C6 893F 214 MOV WORD PTR [BX], DI ;SAVE NEW XPTR				209				
00C1 D1EF 212 SHR DI, 1 00C3 83EB08 213 SUB BX, 8 ;BX = XPTR LOCATION . 00C6 893F 214 MOV WORD PTR [BX], DI ;SAVE NEW XPTR	OORF	:		210	SAVE_N	EW_XPTR:		
00C1 D1EF 212 SHR DI, 1 00C3 83EB08 213 SUB BX, 8 ;BX = XPTR LOCATION . 00C6 893F 214 MOV WORD PTR [BX], DI ;SAVE NEW XPTR	OOBF	D1EF		211		SHR	DI, 1	;DI = DI / 4
00C3 83EB08 213 SUB BX, 8 ;BX = XPTR LOCATION . 00C6 893F 214 MOV WORD PTR [BX], DI ;SAVE NEW XPTR	0001	DIEF						
. 00C6 893F 214 MOV WORD FTR [BX], DI ;SAVE NEW XPTR								BX = XPTR LOCATION
	,	· · · · ·				-		

8085/87/88/185 MACRO	ASSEMBLER	RFUN1S				08/30/84	PAGE	5
LOC OBJ	LINE	SOURCE	Ī					
0008	216	EPIL00	iUE:					
00C8 5D	217		POP	BP	RESTORE REGISTERS			
00C9 C20100	218		RET	1				
	219							
	220	NEARCO)DE	END?				
	221							
	222	CODE	ENDS					
	223							
	224	; ****	*****	**********	**************************************	*****		
	225	; ****	******	********	ዸ፟ፙቔቔቔቔቔቔቔቔቔቔቔቔቔቔቔቔቔቔቔቔቔቔቔቔቔቔቔቔቔቔቔቔቔቔቔቔ	****		
	226							
	227		END					

ASSEMBLY COMPLETE, NO ERRORS FOUND

8086/87/88/186 MACRO ASSEMBLER RFUNIS

SERIES-III 8086/87/88/186 MACRO ASSEMBLER V2.0 ASSEMBLY OF MODULE RFUNIL OBJECT MODULE PLACED IN :F1:RFUNIL.OBJ ASSEMBLER INVOKED BY: ASM86.86 :F1:RFUNIL.SRC

LOC OBJ LINE SOURCE	
1 ; 444444448444444444444444444444444444	·*
• • • • • • • • • • • • • • • • • • • •	 *
	*
	*
, , ,	*
	*
	*
8 ; ***************************	ł*
9 ;	
10 ;	
11 ;	
12 ; THIS SUBROUTINE OUTPUTS THE ESTIMATED VALUE OF A	FUNCTION $YV = F(XV)$
13 ; BY INTERPOLATING FROM KNOWN SOLUTION POINTS, USI	ING THE RELATION
14 ; $YV = ((XV-XL)/(XH_XL))(YL-YL) + YL WHERE$	
15 ; YV = ANSHER TO BE COMPUTED	
16 ; XV = INPUT VALUE	
17 ; XH = KNOWN HIGH X POINT	
18 ; XL = KNOWN LOW X POINT	
19 ; YH = KNOHN HIGH Y POINT	
20 ; YL = KNOWN LOW Y POINT	
21 ;	
22 ; INPUT REQUIREMENTS:	
23 ; 1) THE FOLLOHING ADDRESSES WILL BE PASSED T	TO THIS SUBROUTINE:
24 ; A. ON TOP OF THE 8026 STACK:	
25 ; THE ADDRESS (OFFSET THEN SEGMENT) OF	F A DATA AREA WITH
26 ; THE FOLLOWING FORMAT	
27 ;	
28 DSTRUC STRUC	
29	
0000 30 XPTR DW ? ;X ARRAY	
0002 31 ZPTR DW ? ;Y ARRAY	
	ELEMENTS IN X ARRAY
	ELEMENTS IN Y ARRAY
34 ; XARRAY DD NXPTS DUP(?) ;X ARRAY	
35 ; YARRAY DD NYPTS DUP(?) ;Y ARRAY	Υ
36	
37 DSTRUC ENDS	
38 ;	
39 ; B. 2ND FROM THE TOP OF THE 8086 STACK:	UTOU HOLDO THE V
40 ; THE ADDRESS OF THE MEMORY LOCATION W	utou uncha luc v
41 ; INPUT VALUE 42 ;	
	3 EMPTY REGISTERS
43 ; 2) IT IS REQUIRED THAT THE 8087 CHIP HAVE : 44 ; WHEN THIS INTERPOLATION ROUTINE IS:	
	ONEELO:
4ን :	
45 ; 46 : ONTENT FEFFCTS:	
46 ; OUTPUT EFFECTS:	, SI
46 ; OUTPUT EFFECTS: 47 ; 1) REGISTERS DESTROYED: AX, BX, CX, DX, DI	
46 ; OUTPUT EFFECTS: 47 ; 1) REGISTERS DESTROYED: AX, BX, CX, DX, DI	8087 STACK

```
LOC OBJ
                  LINE
                         SOURCE
                    51
                         ;
                              EXAMPLE FORTRAN CALL:
                    52
                         ;
                    53
                                    SUBROUTINE RTEST
                    54
                                    DIMENSION XARRAY(7), YARRAY(7)
                    55
                                    INTEGER XINDEX, YINDEX
                    56
                                    COMMON /STUFF/XINDEX,ZINDEX,NXPTS,NZPTS,XARRAY,YARRAY
                                    DATA XINDEX,ZINDEX,NXPTS,NZPTS/0,0,7,7/
                    57
                                    DATA XARRAY/0.0.1.0.2.0.3.0.4.0.5.0.6.0/
                    58
                    59
                         ;
                                    DATA YARRAY/0.0,1.0,4.0,9.0,16.0,25.0,36.0/
                    60
                                      X=1.5
                    61
                         i
                                      Y=RFUNIL(X,XINDEX)
                    62
                                      PRINT *, Y
                    63
                                    END
                    64
                    65
                        66
                    67
                    68
                              NAME
                                   RFUNIL
                    69
                        CGROUP GROUP
                                   CODE
                    70
                        DGROUP GROUP
                                   DATA
                    71
                              ASSUME CS:CGROUP, DS:DGROUP, ES:DGROUP
                    72
                              PUBLIC RFUNIL
                    73
                    74
                        75
                    76
                              STACK STRUCTURE
                    77
                   78
                        PARAMS STRUC
                   79
0000
                   80
                        OLD_ES DW
                                         ;SAVED ES REGISTER
0002
                   81
                        OLD_BP DW
                                         SAVED BP REGISTER
0004
                        OLD_DS DW
                   82
                                         SAVED DS REGISTER
0006
                        RETURN DD
                   83
                                   2
                                         RETURN ADDRESS (OFFSET ON TOP, THEN SEGMENT)
000A
                   84
                        DADDR DD
                                   ?
                                         ;ADDRESS OF DATA STRUCTURE (OFFSET, THEN SEGMENT)
000E
                   85
                        ΧV
                              DD
                                   ?
                                         (ADDRESS OF XV (OFFSET, THEN SEGMENT)
                   86
                   37
                        PARAMS ENDS
                   88
                   89
                        90
                        91
                   92
                        DATA SEGMENT PUBLIC 'DATA'
                   93
0000 ????
                   94
                        STYSAVE DW
                   95
                   96
                        DATA
                             ENDS
                   97
                   98
                        99
                        100
                  101
                        CODE
                             SEGMENT PUBLIC 'CODE'
                  102
0000
                  103
                        FARCODE PROC
                                   FAR
                  104
                  105
```

LOC OBJ		LINE	SOURCE			
0000 0000 1E 0001 55 0002 06 0003 8BEC		106 107 108 109 110	RFUNIL:	PUSH PUSH PUSH MOV	DS BP ES BP, SP	SAVE REGISTERS
0005 B8 0008 8EC0 000A C55E0E 000D 9BD907	R	112 113 114 115		MOV MOV LDS FLD LDS	AX, DATA ES, AX BX, [BP].XV DHORD PTR [BX] BX, [BP].DADDR	GET XV ADDRESS AND SEGMENT PUT XV ON TOP OF 8087 STACK GET START ADDRESS OF DATA STRUCTURE
0010 C55E0A 0013 8B3F 0015 D1E7 0017 D1E7 0019 8B7704		116 117 118 119 120		MOV SHL SHL MOV	DI, [BX].XPTR DI, 1 DI, 1 SI, [BX].NXPTS	DI = ELEMENT INDEX DI = BYTE INDEX = ELEMENT INDEX * 4 SI = NXPTS
001C D1E6 001E D1E6 0020 83C308 0023 8BD3 0025 03D6		121 122 123 124 125		SHL SHL ADD MOV ADD	SI, 1 SI, 1 BX, 8 DX, BX DX, SI	;SI = NO. OF BYTES=NXPTS * 4 ;BX = START OF X ARRAY ;DX = START OF Y ARRAY
0027 9BD311 002A 9B26DD3E0000 0030 9B 0031 26A10000	R R	126 +2 127 +1 128 +1 129 +1		FCOM FSTSW FWAIT MOV	DWORD FTR (BX + DI) ES:STWSAVE AX, ES:STWSAVE	; COMPARE ARRAY ELEMENT WITH XV
0035 250041 0038 3B0001 003B 7434		130 +1 131 +2 132 +2 133 +2 134 +1		and CMP JE	AX, 4100H AX, 0100H LEFT_OF_START	; MASK-IN COMPARISON BITS ; EXIT IF XV < ARRAY ELEMENT
003D 003D 83C704 0040 3BFE 0042 7420		135 136 137 138 139	RIGHT_0	DF_START: ADD CMP JE	DI, 4 DI, SI L1	; IF X > DATA(ORIGINAL INDEX) ;*** THEN BEGIN *** ; INDEX = INDEX + 4 ; WHILE (INDEX <= NO. OF BYTES) ;
0044 9BD811 0047 9B26DD3E0000 004D 9B 004E 26A10000	R R	140 141 +2 142 +1 143 +1 144 +1		FCOM FSTSW FWAIT MOV	DWORD PTR (BX + DI) ES:STWSAVE AX, ES:STWSAVE	; *** DO BEGIN *** ; COMPARE ARRAY ELEMENT WITH XV ; MASK-IN COMPARISON BITS
0052 250041 0055 3D0001 0058 740A 005A 3D0040 005D 7469		145 +1 146 +2 147 +2 148 +2 149 +2		and CHP JE CHP JE	AX, 4100H AX, 0100H L1 AX, 4000H BREAKPOINT	<pre>; mask-in comparison bits ; EXIT IF XV < ARRAY ELEMENT ; EXIT IF XV = ARRAY ELEMENT</pre>
005F 83C704 0062 EBDC 0064 83EE04 0067 83EF04		150 +1 151 152 153 154 155	Li:	add Jmp Sub Sub Cmp	DI, 4 RS1 SI, 4 DI, 4 DI, SI	; INDEX = INDEX + 4 ; *** END (WHILE) *** ; ; ; IF INDEX > NO. OF BYTES
006A 3BFE 006C 745A 006E EB3490		156 157 158 159 160	LEFT_(JE JMP DF_START:	OUT_OF_RANGE FOUND_LOW_INDEX	; THEN OUT_OF_RANGE ; ELSE FOUND_LOW_INDEX ;*** END (IF) *** ;IF X < DATA(ORIGINAL INDEX) ;*** THEN BEGIN ***

LOC OBJ	LIN	E SOURCE			
0071 83EF04	161	İ	SUB	DI, 4	. TAIDEN - YAIDEN A
0074 83FF00	163		CMP	DI, 0	; INDEX = INDEX - 4 ; WHILE (INDEX > 0)
0077 7C20	163		JL.	MÍ	; while (index > 0)
••••	164		02	111	, ; *** DO BEGIN ***
0079 9BD811		5 +2	FCOM	DWORD PTR [BX + DI]	; COMPARE ARRAY ELEMENT WITH XV
007C 9B26DD3E0000		5 +1	FSTSW	ES: STWSAVE	A COURTER UNION FEETILISM MILLI MA
0082 9B		7 +1	FUAIT		
0033 26A10000		3 +1	MOV	AX, ES:STWSAVE	
0087 250041	169	+1	AND	AX, 4100H	; MASK-IN COMPARISON BITS
008A 3D0000	170) +2	CMP	AX, 0000H	: EXIT IF XV > ARRAY ELEMENT
008D 740A	171	+2	JE	M1	
008F 3D0040		2 +2	CMP	AX- 4000H	
0092 7434		3 +2	JE	BREAKPOINT	; EXIT IF XV = ARRAY ELEMENT
		+1			
0094 83EF04	175		SUB	DI, 4	: INDEX = INDEX - 4
0097 EBDB	176		JPP OMP	LS1	; *** END (WHILE) ***
0099 83FF00 009C 7D06	177		CMP	DI, O	; IF INDEX ⇔ 0
009E 83C704	178		JGE	FOUND_LOW_INDEX	; THEN FOUND_LOW_INDEX
009E 83C704 00A1 EB2590	179 180		ADD	DI, 4	5 CLOS OUT OF DAVID
VVH1 EB2J70	181		JMP	OUT_OF_RANGE	: ELSE OUT_OF_RANGE;
	182				;*** END(ID) ***
00A4	183		LOW_INDEX	٧•	
00A4	184			۸•	
00A4 9BD901	185		FLD	DWORD PTR [BX+DI]	STORE XL ON STACK TOP
00A7 9BDCE9	186		FSUB	ST(1), ST	;(XV-XL) ON STACK BOTTOM
00AA 9BD86904	187		FSUBR	DWORD PTR [BX+DI+4]	;(XH-XL) ON STACK TOP
OOAE 9BDEF9	188		FDIV	Emerica Control Control	(XV-XL)/(XH-XL) LEFT ON STACK
00B1 87DA	189	•	XCHG	BX, DX	THE REPORT REPORTED TO GRADIN
00B3 9BD901	190)	FLD	DWORD PTR (BX+DI)	STORE YL ON STACK TOP
00B6 9BD94104	191		FLD	DWORD PTR [BX+DI+4]	STORE YH ON STACK TOP
00BA 9BD8E1	192	<u>:</u>	FSUB	ST, ST(1)	;(YH-YL) ON STACK TOP
OOBD 9BDECA	193		FMULP	ST(2), ST	:((XV-XL)/(XH-XL))(YH-YL)ON STACK BOT
00CO 9BDEC1	194		FADD		; YV ON STACK TOP (ANSWER)
00C3 87DA	195		XCHG	BX, DX	
00C5 EB0B90	196		JMP	SAVE_NEW_XPTR	
0008	197				
0008	198				
00C8 9BDDD8	199			OT (A)	
00CB 87D3	200 201		FSTP XCHG	ST(0)	DISCARD TOP OF STACK
00CD 9BD901	201		FLD	DX, BX DWORD PTR [BX + DI]	*CARC DOMINOU V HALES
00D0 87D3	203		XCHG	DX, BX	SAVE BOUNDRY Y VALUE
	204		NO 10	DAT DA	
00D2	205		W_XPTR:		
00D2 D1EF	206		SHR	DI, 1	;DI = DI / 4
00D4 D1EF	207		SHR	DI, 1	722 21 1
00D6 83EB08	208		SUB	BX, 8	BX = XPTR LOCATION
00D9 893F	209		VOM	WORD PTR [BX], DI	SAVE NEW XPTR
	210				
OODB	211	-	E:		
00DB 07	212		P0P	ES	
00DC 5D	213		POP	BP	RESTORE REGISTERS
OODD 1F	214		POP	DS	
00DE CA0800	215		RET	8	

LOC OBJ	LINE	SOURCE	
	216		
	217	FARCODE	ENDP
	218		
	219	CODE	ENDS
	220		
	221	;*****	***************************************
	222	; *** ** *	**************
	223		
	224		END

08/30/84 PAGE 5

ASSEMBLY COMPLETE, NO ERRORS FOUND

8086/87/88/186 MACRO ASSEMBLER RFUNIL

SERIES-III 8086/87/88/186 MACRO ASSEMBLER V2.0 ASSEMBLY OF MODULE RFUN2S OBJECT MODULE PLACED IN :F1:RFUN2S.OBJ ASSEMBLER INVOKED BY: ASM86.86 :F1:RFUN2S.SRC

LOC OBJ	LINE	SOURCE	
	i	;	************
	2	;	* ROUTINE NAME: RFUN2S *
	3	;	* DATE: AUG. 27, 1983 *
	4	;	* PROGRAHMER'S NAME: MICHAEL MACKIN *
	5	;	* PURPOSE: TO INTERPOLATE A SIMPLE *
	6	;	* FUNCTION FOR SMALL-MODEL *
	7	;	* PROGRAMS *
	8 9	; •	#*#***********************************
	10	,	THIS CUMPONITIES ON THE FORWARD HALVE OF A PRACTICAL TO
	11	•	THIS SUBROUTINE OUTPUTS THE ESTIMATED VALUE OF A FUNCTION YV = F(XV, ZV) BY INTERPOLATING FROM KNOWN SOLUTION POINTS, USING THE RELATIONS
	12	;	YL = ((XV-XL)/(XH-XL))(YB-YA) + YA
	13	•	YH = ((XV-XL)/(XH-XL))(YD-YC) + YC
	14	;	YV = ((ZV-ZL)/(ZH_ZL))(YH-YL) + YL
	15	;	
	16	;	INPUT REQUIREMENTS:
	17	;	 THE FOLLOWING ADDRESSES WILL BE PASSED TO THIS SUBROUTINE:
	18	i	A. ON TOP OF THE 8086 STACK:
	19	;	THE ADDRESS (OFFSET) OF A DATA AREA WITH THE FOLLOWING
	20	;	FORMAT
	21	;	EATONA ATONA
	22 2 3		DSTRUC STRUC
0000	23 24		XPTR DN ? :X ARRAY INDEX
0002	25		TO THE STEEL
0004	26		ZPTR DW ? ;Z ARRAY INDEX NXPTS DW ? ;NO. OF ELEMENTS IN X ARRAY
0006	27		NZPTS DW ? INO. OF ELEMENTS IN Z ARRAY
	28	;	ZARRAY DD NZPTS DUP(?) ; Z ARRAY (255 ELEMENTS MAXIMUM)
	29	;	XARRAY DD NXPTS DUP(?) ;X ARRAY
	30	;	YARRAY DD NYPTS DUP(?) ;Y ARRAY
	31		
	32		DSTRUC ENDS
	33	•	
	34 35	;	B. ON THE 8087 STACK:
	35 36	;	ON TOP ZV, THEN XV
	37	•	2) IT IS DECRIBED THAT THE GOOD CHIR HAVE E CHICAGO PERSONNEL
	38	;	 IT IS REQUIRED THAT THE 8087 CHIP HAVE 5 EMPTY REGISTERS WHEN THIS INTERPOLATION ROUTINE IS CALLED.
	39	;	when this intensional registre is Cheleb.
	40	;	3) NOTE: NO THO ADJACENT ELEMENTS OF XARRAY OR YARRAY MAY HAVE
	41	;	IDENTICAL ELEMENTS. THIS WILL RESULT IN DIVISION BY ZERO.
	42	;	THE THE PARTICION OF LENOT
	43	;	OUTPUT EFFECTS:
	44	;	1) REGISTERS DESTROYED: AX, BX, CX, DX, DI, SI
	45	;	2) INTERPOLATED RESULT RETURNED ON TOP OF 8087 STACK
	46 47	; •	3) XPTR LOCATION UPDATED TO INDEX X VALUE JUST PRECEDING XV.
	47 48	; ;	ZPTR LOCATION UPDATED TO INDEX Z VALUE JUST PRECEDING ZV. (TO SPEED FUTURE SEARCHES FOR X AND Z)
	49	;	(10 SPEED FOUNDE SERNORES FOR X AND Z)
	50	;	EXAMPLE PL/M CALL:

2

```
LOC OBJ
                      LINE
                               SOURCE
                        51
                                      $SMALL
                         52
                               i
                         53
                              ;
                         54
                                      TSTPGM: DO:
                         55
                                      /* THIS IS A PROGRAM TO TEST ROUTINE RFUN2S */
                         56
                         57
                         58
                                             DECLARE DSTRUC STRUCTURE (
                         59
                                                XINDEX INTEGER, ZINDEX INTEGER, NXPTS INTEGER, NZPTS INTEGER,
                                               ZARRAY(3) REAL, XARRAY(5) REAL, YARRAY(15) REAL) PUBLIC
                         60
                                                INITIAL (
                         61
                                                  /* XINDEX */ 0,
                         62
                                                  /* ZINDEX */ 0,
                         63
                               ;
                                                  /* NXPTS */ 5,
                         64
                         65
                                                  /* NZPTS */ 3,
                               ÷
                         66
                                                  /* ZARRAY */ 0.0, 10.0, 20.0,
                                                  /* XARRAY */ 0.0, 2.0, 4.0, 6.0, 8.0,
                         67
                               i
                                                  /* YARRAY */ 0.0, 2.0, 4.0, 6.0, 8.0,
                         68
                                                              10.0, 12.0, 14.0, 16.0, 18.0,
                         69
                         70
                                                              20.0, 22.0, 24.0, 26.0, 28.0 );
                         71
                               i
                                             RFUNZS: PROCEDURE(X,Z,STRUC_ADDR) REAL EXTERNAL;
                         72
                                                    DECLARE X REAL;
                         73
                                                    DECLARE Z REAL;
                         74
                               ;
                                                    DECLARE STRUC_ADDR POINTER;
                         75
                               ÷
                                             END RFUN2S;
                         76
                               ;
                         77
                               ţ
                                             FUNPGM: PROCEDURE PUBLIC;
                         78
                         79
                                                    DECLARE I INTEGER;
                         80
                                                    DECLARE (X,Z,Y) REAL;
                         81
                                                       X = 1.5;
                         82
                                                       I = 12.0;
                         83
                                                       Y = RFUN2S(X, Z, @DSTRUC);
                                                       CALL FRINT_ANSWER;
                         84
                                                    END;
                         85
                               ï
                                             END FUNPON;
                         86
                               ţ
                         87
                                      END TSTPGM;
                         83
                         89
                         90
                               91
                         92
                                             RFUN2S
                         93
                                      NAME
                         94
                               CGROUP GROUP CODE
                         95
                               DGROUP GROUP DATA
                                      ASSUME CS:CGROUP, DS:DGROUP
                         96
                         97
                                      PUBLIC RFUN2S
                         98
                         99
                               100
                        101
                                      STACK STRUCTURE
                        102
                               PARAMS STRUC
                        103
                        104
 0000
                        105
                               OLD_BP DW
                                             ?
                                                   SAVED BP REGISTER
```

3

```
LOC OBJ
                       LINE
                               SOURCE
 0002
                        106
                               RETURN DW
                                             ?
                                                    RETURN ADDRESS
 0004
                        107
                               DADDR
                                      DW
                                             2
                                                    ; ADDRESS OF DATA STRUCTURE
                        103
                        109
                               PARAMS ENDS
                        110
                        111
                               112
                               <del></del>
                        113
                        114
                               DATA
                                      SEGMENT PUBLIC 'DATA'
                        115
 0000 ????
                        116
                               STWSAVE DW
                        117
                        118
                               DATA
                                      ENDS
                        119
                        120
                               121
                               <del>┇╫╫╫╫╫╫╫╫╫╫╫╫╫╫╫╫╫╫</del>
                        122
                        123
                               CODE
                                      SEGMENT PUBLIC 'CODE'
                        124
0000
                        125
                              NEARCODE
                                            PROC
                                                   NEAR
                        126
                        127
                        128
0000
                        129
                              RFUN2S:
                        130
                                                                 ;SAVE REGISTERS
0000 55
                        131
                                     PUSH
                                            BP
0001 8BEC
                        132
                                     MOV
                                            BP, SP
0003 8B5E04
                        133
                                     YÜK
                                            BX, [BP].DADDR
                                                                 :GET START ADDRESS OF DATA STRUCTURE
0006 8BD3
                        134
                                     HOV
                                            DX, BX
                                                                 ; SAVE BX
0008 890F
                       135
                                     HOV
                                            CX, [BX].XPTR
                                                                 ;CX = XPTR
000A 8B6F04
                       136
                                     VOM
                                            BP, [BX].NXPTS
                                                                 BP = NXPTS
000D 8B7F02
                        137
                                     YOM
                                            DI, [BX].ZPTR
                                                                 ;DI = ZPTR
0010 D1E7
                       138
                                     SHL
                                            DI, 1
0012 D1E7
                       139
                                            DI, 1
                                     SHL
                                                                 ;DI = BYTE INDEX = ELEMENT INDEX * 4
0014 8B7706
                       140
                                     MOV
                                            SI, [BX].NZPTS
                                                                 ;SI = NZPTS
0017 D1E6
                       141
                                     SHL
                                            SI, 1
0019 D1E6
                       142
                                     SHL
                                            SI, 1
                                                                 ;SI = NO. OF BYTES = NZPTS * 4
                       143
G01B
                       144
                              GET_Z_INDEX:
001B 83C308
                       145
                                     ADD
                                            BX, 8
                                                                 BX = START OF Z ARRAY
001E 9BD811
                       146 +2
                                     FCOM
                                            DWORD PTR [8X + DI]
                                                                      COMPARE ARRAY ELEMENT WITH ZV
                                                                 ;
0021 9BDD3E0000
                       147 + 1
                                     FSTSN
                                            STUSAVE
0026 9B
                       143 + 1
                                     FWAIT
0027 A10000
                       149 +1
                                     VON
                                            AX, STHSAVE
002A 250041
                       150 +1
                                     AND
                                            AX, 4100H
                                                                       MASK-IN COMPARISON BITS
002D 3D0001
                       151 +2
                                     CHP
                                            AX, 0100H
0030 7432
                       152 +2
                                     JΕ
                                            LEFT_OF_Z_START
                                                                                    EXIT IF ZV C ARRAY ELE
                              MENT
                       153 +2
                       154
                                                                 ; IF Z > DATA(ORIGINAL INDEX)
0033
                       155
                              RIGHT_OF_Z_START:
                                                                 **** THEN BEGIN ***
0032 830704
                       156
                                     ADD
                                            DI, 4
                                                                 ; INDEX = INDEX + 4
0035 3BFE
                       157
                              RZS1:
                                     CMP
                                            DI, SI
                                                                    WHILE (INDEX <= NO. OF BYTES)
0037 7419
                       158
                                     JĖ
                                            RZS2
                       159
                                                                       *** DO BEGIN ***
```

LOC OBJ		LINE	SOURCE			
0039 9BD811 003C 9BDD3E0000 0041 9B	R	160 +2 161 +1 162 +1		FCOM FSTSW FWAIT	DWORD PTR IBX + DII STWSAVE	; COMPARE ARRAY ELEMENT WITH ZV
0042 A10000	R	163 +1 164 +1		MOV AND	AX, STWSAVE AX, 4100H	; MASK-IN COMPARISON BITS
0045 250041 0048 300001		165 +2		CHID	AX, 0100H) Andy In Odd Attoon Pilo
004B 7405		165 +2 167 +2		JE	RZS2	; EXIT IF ZV < ARRAY ELEMENT
004D 83C704		168		ADD	DI, 4	; $INDEX = INDEX + 4$
0050 EBE3		169	B700.	JMP	RZS1	; *** END (WHILE) ***
0052 8BC6		170	RZS2:	MOV SUB	AX, SI AX, 4	;
0054 2D0400 0057 83EF04		171 172		SUB	DI, 4	;
005A 3BF8		172		CMP	DI, AX	; IF INDEX \bigcirc NO. OF BYTES
005C 7540		174		JNE	FOUND_LOW_Z_INDEX	; THEN FOUND_LOW_INDEX
005E 9BDDD3		175		FSTP	ST(0)	; ELSE OUT_OF_RANGE
0061 EB4B90		176		JWP	GET_X_INDEX	;
		177				;*** END (IF) ***
0014		178	1 FET 0	T OTAD	Ť:	; IF Z < DATA(ORIGINAL INDEX) ;*** THEN BEGIN ***
0064 0064 83EF04		179 180	LEF 1_U	=_z_star: Sub	DI, 4	; INDEX = INDEX - 4
0067 83FF00		181	LZS1:	CMP	DI, 0	; WHILE (INDEX > 0)
006A 7C1E		182	2201.	JL	LZS2	3
••••		183				; *** DO BEGIN ***
006C 9BD911		184 +2		FCOM	DWORD PTR [BX + DI]	; COMPARE ARRAY ELEMENT WITH ZV
006F 9BDD3E0000	R	185 +1		FSTSW	STVISAVE	
0074 9B	_	186 +1		FWAIT	AV OTHORIE	
0075 A10000	R	187 +1 183 +1		NOV AND	AX, STWSAVE AX, 4100H	; MASK-IN COMPARISON BITS
0078 250041 007B 3D0000		169 +2		CMP	AX, 0000H	; EXIT IF ZV > ARRAY ELEMENT
007E 740A		190 +2		JE	LZS2) DALL II LT 2 (IIIIII) Limitely
0080 3D0040		191 +2		CMP	AX, 4000H	
0083 7405		192 +2		JΕ	LZS2	; EXIT IF ZV = ARRAY ELEMENT
0085 83EF04		193		SUB	DI, 4	; INDEX = INDEX - 4
0088 EBDD		194	1.700+	JMP	LZS1	; *** END (WHILE) ***
008A 83FF00		195	LZS2:	CMP JGE	DI, O FOUND_LOW_Z_INDEX	; IF INDEX <> 0 ; THEN FOUND_LOW_INDEX
008B 7D0F 008F 83C704		196 197		ADD	DI, 4	; ELSE OUT_OF_RANGE
CO92 9BDDD8		198		FSTP	ST(0)	;
0095 98D9EE		199		FLDZ		; REPLACE TOP WITH O
0098 9BD9C9		200		FXCH	ST(1)	; PUT XV ON TOP OF 8087 ST.
009B EB1190		201		JMP	GET_X_INDEX	;
,		202 203				;*** END(IF) ***
009E		204	FOUND_	TOMTZTIV		ACTOR THE ON OTAGE TOR
009E 9BD901		205		FLD	DWORD PTR [BX+DI]	;STORE ZL ON STACK TOP ;(ZV-ZL) 2ND FROM TOP
00A1 9BDCE9 00A4 9BD86904		206 207		FSUB FSUBR	ST(1), ST DWORD PTR [BX+DI+4]	;(ZH-ZL) ON STACK TOP
00A8 9BBEF9		207		FDIV	DUCKE I III EBV. DI. 41	;(ZV-ZL)/(ZH-ZL) LEFT ON STACK
00AB 9BD9C9		209		FXCH	ST(1)	PUT XV ON TOP OF 8037 STACK
00AE		210 211	GET Y	INDEX:		
OOAE O3DE		212	051242	ADD	BX, SI	;BX = START OF XARRAY
00B0 87F9		213		XCHG	DI, CX	:DI = XPTR : CX = LOW_Z_INDEX
00B2 D1E7		214		SHL	DI, 1	

LOC OBJ		LINE	SOURCE				
00B4 D1E7 00B6 87F5 00B8 D1E6 00BA D1E6 00BC 9BD811 00BF 9BDD3E0000 00C4 9B 00C5 A10000	P.	215 216 217 218 219 +2 220 +1 221 +1 222 +1		SHL XCHG SHL SHL FCOM FSTSW FWAIT MOV	DI, 1 SI, BP SI, 1 SI, 1 DWORD PTR (BX + DI) STWSAVE AX, STWSAVE	;SI	= BYTE INDEX = XPTR * 4 = NXPTS = NO. OF X BYTES = NXPTS * 4 COMPARE ARRAY ELEMENT WITH XV
00C8 250041 00CB 3D0001 00CE 7432	••	223 +1 224 +2 225 +2	HENT	AND CMP JE	AX, 4100H AX, 0100H LEFT_OF_X_START	;	MASK-IN COMPARISON BITS ; EXIT IF XV < ARRAY ELE
00D0 00D0 83C704 00D3 3BFE 00D5 7419		226 +2 227 228 229 230 231 232 233 +2	RIGHT_(DF_X_STA ADD CMP JE FCOM	RT: DI, 4 DI, SI RXS2 DWORD PTR (BX + DI)	;**:	X > DATA(ORIGINAL INDEX) THEN BEGIN *** INDEX = INDEX + 4 WHILE (INDEX <= NO. OF BYTES) *** DO BEGIN *** COMPARE ARRAY ELEMENT HITH XV
00DA 9BDB3E0000 00DF 9B 00E0 A10000 00E3 250041 00E6 3D0001 00E9 7405	R	234 +1 235 +1 236 +1 237 +1 238 +2 239 +2 240 +2		FSTSN FWAIT MOV AND CMP JE	STWSAVE AX, STWSAVE AX, 4100H AX, 0100H RXS2	;	MASK-IN COMPARISON BITS EXIT IF XV < ARRAY ELEMENT
00EB 83C704 00EE EPE3 00F0 8BC6 00F2 2D0400 00F5 83EF04 00F8 3BF8 00FA 753D 00FC 9BDDB 00FF EB4590		241 242 243 244 245 246 247 248 249	RXS2:	ADD JMP MGV SUB SUB CMP JME FSTP JMP	DI, 4 RXS1 AX, SI AX, 4 DI, 4 DI, AX FOUND_LOW_X_INDEX ST(0) GET_YL	;	INDEX = INDEX + 4 *** END (WHILE) *** IF INDEX <> NO. OF BYTES THEN FOUND_LOW_INDEX ELSE OUT_OF_RANGE END (IF) ***
0102 0102 83EF04 0105 83FF00 0108 7C1E 010A 9BD811 010D 9BDB3E0000	R	251 252 253 254 255 256 257 +2 258 +1	LEFT_OF	LXLSTAR1 SUB CMP JL FCOH FSTSW	DI, 4 DI, 0 LXS2 DWORD PTR [BX + DI] STWSAVE	; IF	X < DATA(ORIGINAL INDEX) THEN BEGIN *** INDEX = INDEX - 4 WHILE (INDEX > 0) *** DO BEGIN *** COMPARE ARRAY ELEMENT WITH XV
0112 9B 0113 A10000 0116 250041 0119 3D0000 011C 740A 011E 3D0040 0121 7405 0123 83EF04 0126 EBDD 0128 83FF00	R	259 +1 260 +1 261 +1 262 +2 263 +2 264 +2 265 +2 266 267	LXS2:	FWAIT MOV AND CMP JE CMP JE SUB JMP CMP	AX, STWSAVE AX, 4100H AX, 0000H LXS2 AX, 4000H LXS2 BI, 4 LXS1 BI, 0	;	MASK-IN COMPARISON BITS EXIT IF XV > ARRAY ELEMENT EXIT IF XV = ARRAY ELEMENT INDEX = INDEX - 4 *** END (WHILE) *** IF INDEX <> 0

LOC OBJ	LINE	SOURCE		
012B 7D0C	269	JGE	FOUND_LOW_X_INDEX	; THEN FOUND_LOW_INDEX
012D 83C704	270	ADD	DI, 4	; ELSE OUT_OF_RANGE
0130 9BDDD8	271	FSTP	ST(0)	1
0133 9BD9EE	272	FLDZ		; REPLACE TOP WITH O
0136 EB0E90	273	JMP	GET_YL	;
	274			:*** END(IF) ***
	275			
0139	276	FOUND_LGW_X_IN	GEX:	
0139 9BD901	277	FLD	DWORD PTR [BX+DI]	STORE XL ON STACK TOP
013C 9BDCE9	278	FSUB	ST(1), ST	;(XV-XL) ON STACK BOTTOM
013F 9BD86904	279	FSUBR	DWORD PTR [BX+DI+4]	; (XH-XL) ON STACK TOP
0143 9BDEF9	280	FDIV	2115115 1 111 221 22 72	;(XV-XL)/(XH-XL) LEFT ON STACK
9110 700017	281	1 224		TINY KETTANI KET EEL TON OTHER
0146	231 232	GET_YL:		
0145 03DE	293	ADD	PX, SI	;BX = START OF YARRAY
0148 8BC6	284	MOV	AX, SI	AX = NO. OF X BYTES
014A D1E8	285	SHR	AX, 1	AAV - NVETO - NO OF V EVITO / A
014C D1E8	286	SHR	AX, 1	AX = NXPTS = NO. OF X BYTES / 4
014E F6E1	287	HUL	CL	AX = YA ROW OFFSST
	288			THE AVOVE INSTR. LIMITS LENGTH OF ZARRAY
A4EA A8EA	289			:TO 255 ELEMENTS
0150 03D8	290	ADD	BX, AX	BX = YA ROW PTR = NXPTS * ZPTR
0152 03DF	291	ADD	BX, DI	;BX = YA POINTER
0154 9BD907	292	FLD	DWORD PTR [BX]	STORE YA ON ST TOP
0157 8BC7	293	VOM	AX, DI	
0159 050400	294	ADD	AX, 4	
015C 3BC6	295	CMP	AX, SI	
015E 7410	296	JE	GET_YH	EXIT IF XV PAST BOUNDRY
0160 9BD9C9	297	FXCH	ST(1)	;(XV-XL)/(XH-XL) ON TOP, THEN YA
0163 9BD94704	298	FLD	DWORD PTR [BX + 4]	ISTORE YB ON ST. TOP
0167 9BD8E2	299	FSUB	ST, ST(2)	;(YB-YA) ON ST. TOP
016A 9BD8C9	300	FMUL	ST, ST(1)	;((XV-XL)/(XH-XL))(YB-YA) ON ST TOP
016D 9BDEC2	301	FADDP	ST(2), ST	;YL 2ND FROM TOP OF STACK
	302			
0170	303	GET_YH:		
0170 8BC1	304	VOM	AX, CX	; IF (Z_POINTER TOO BIG)
0172 050400	305	ADD	AX, 4	;*** DO REGIN ***
0175 3BC5	306	CMP	AX, BP	
0177 7 50F	307	JNE	GYH1	
0179 8BC7	308	Vem	AX, DI	; IF (X_POINTER TOO BIG)
017B 050400	309	ADD	AX, 4	
017E 3BC6	310	CMP	AX, SI	
0180 742A	311	JE	FOUND_YV	; THEN EXIT
0182 9BDDD8	312	FSTP	ST	; ELSE DISCARD JUNK
0185 EP2590	313	JMP	FOUND_YV	; AND THEN EXIT;
	314			;*** END (IF) ***
0188 03DE	315	GYH1: ADD	BX, SI	BX = YC POINTER
018A 9BD907	316	FLD	DWORD PTR [BX]	STORE YO ON STACK TOP
018D 8BC7	317	VOH	AX, DI	
018F 050400	318	ADD	AX, 4	
0192 3806	319	CMP	AX, SI	
0194 740D	320	JE	GET_YV	;EXIT IF XV PAST BOUNDRY
0196 9BD94704	321	FLD	DWCRD PTR [BX + 4]	
019A 9BD8E1	322	FSUB	ST, ST(1)	;YD-YC ON ST TOP
019D 9BDECA	323	FMULP	ST(2), ST	;((XV-XL)/(XH-XL))(YD-YC) 2ND FROM TOP
			-:- - :-	The state of the s

L00	0BJ	LINE	SOURCE			
01A0	9BDEC1	324		FADD		; YH ON TOP OF 8087
0100		325	OFT MI			-VAL TOD TURN VIL THEN ALORE
01A3		326	GET_YV:	ESUB	AT AT	YH TOP, THEN YL, THEN SLOPE
	9BD8E1	327		FSUB	ST, ST(1)	;YH-YL ON ST. TOP
	9BDECA	328		FMULP	ST(2), ST	;((ZV-ZL)/(ZH-ZL))(YH-YL) 2ND FROM TOP
91A9	9BDEC1	329		FADD		:YV GN ST. TOP
		330				
OIAC		331	FOUND_Y	٧:		
01AC		332	SAVE_NE	W_POINTE	RS:	
01AC	SBDA	333		MOV	BX, DX	
01AE	DIEF	334		SHR	DI, i	
01B0	DIEF	335		SHR	DI, i	
01B2	893F	336		VOM	EBX3.XPTR, DI	;SAVE XPTR
0184	D1E9	337		SHR	CX, 1	
01B6	D1E9	338		SHR	CX, 1	
01B8	894F02	339		MOV	[BX].ZPTR, CX	SAVE IPTR
		340				
01BB		341	EPIL06U	E:		
01BB	5D	342		POP	BP	;RESTORE REGISTERS
	C20100	343		RET	1	
		344				
		345	NEARCOD	E	ENDP	
		346		-		
		347	CODE	ENDS		
		348				
		349	; ******	******	**********	**************************************
		350	•			*************
		351				
		352		END		
		~~-		_,,_		

ASSEMBLY COMPLETE, NO ERRORS FOUND

SERIES-III 8086/87/88/186 MACRO ASSEMBLER V2.0 ASSEMBLY OF MODULE RFUNZL OBJECT MODULE PLACED IN :F1:RFUN2L.OBJ ASSEMBLER INVOKED BY: ASM86.86 :F1:RFUN2L.SRC

LOC OBJ	LINE	SOURCE	
	1	;	*******
	2	;	* ROUTINE NAME: RFUNZL *
	3	;	* DATE: SEP. 1, 1983 *
	4	;	* PROGRAMMER'S NAME: HICHAEL MACKIN *
	5	;	* PURPOSE: TO INTERPOLATE A SIMPLE *
	6	;	* FUNCTION FOR LARGE-MODEL *
	7	;	* PROGRAMS *
	3	•	**************************************
	ģ	;	
	10	į	THIS SUBROUTINE OUTPUTS THE ESTIMATED VALUE OF A FUNCTION YV = F(XV,ZV)
	11	•	BY INTERPOLATING FROM KNOWN SOLUTION POINTS, USING THE RELATIONS
	12	;	YL = ((XV-XL)/(XH-XL))(YE-YA) + YA
	13	;	YH = ((XV-XL)/(XH-XL))(YD-YC) + YC
	14	;	YV = ((ZV-ZL)/(ZH-ZL))(YH-YL) + YL
	15	;	
	16	;	INPUT REQUIREMENTS:
	17	;	 THE FOLLOWING ADDRESSES WILL BE PASSED TO THIS SUBROUTINE:
	18	;	A. ON TOP OF THE 8086 STACK:
	19	;	THE ADDRESS (OFFSET) OF A DATA AREA WITH THE FOLLOWING
	20	ţ	FORMAT
	21	;	
	22		DSTRUC STRUC
	23		
0000	24		XPTR DN ? ;X ARRAY INDEX
0002	25		ZPTR DW ? ;Z ARRAY INDEX
0004	26		NXPTS DW ? ;NO. OF ELEMENTS IN X ARRAY
0006	27		NZPTS DW ? ;NO. OF ELEMENTS IN Z ARRAY
	28	•	ZARRAY DD MZPTS DUP(?) ;Z ARRAY (255 ELEMENTS MAXIMUM)
	29	;	XARRAY DD MXPTS DUP(?) ;X ARRAY
	30	;	YARRAY DD - NYPTS DUP(?) ;Y ARRAY
	31		
	32		DSTRUC ENDS
	33	;	
	34	;	B. NEXT ON THE 8086 STACK:
	35	;	THE ADDRESS OF THE MEMORY LOCATION WHICH HOLDS THE X
	36	;	INPUT VALUE.
	37	;	
	38	;	 IT IS REQUIRED THAT THE 8087 CHIP HAVE 5 EMPTY REGISTERS
	39	;	WHEN THIS INTERPOLATION ROUTINE IS CALLED.
	40	ţ	
	41	;	3) NOTE: NO TWO ADJACENT ELEMENTS OF XARRAY OR YARRAY MAY HAVE
	42	;	IDENTICAL ELEMENTS. THIS WILL RESULT IN DIVISION BY ZERO.
	43	;	
	44	;	OUTPUT EFFECTS:
	45	ï	1) REGISTERS DESTROYED: AX, BX, CX, DX, DI, SI
	46	i	2) INTERPOLATED RESULT RETURNED ON TOP OF 8087 STACK
	47	;	3) XPTR LOCATION UPDATED TO INDEX X VALUE JUST PRECEDING XV.
	48	5	ZPTR LOCATION UPDATED TO INDEX Z VALUE JUST PRECEDING ZV.
	49	;	(TO SPEED FUTURE SEARCHES FOR X AND Z)
	50	;	

2

```
LOC OBJ
                    LINE
                           SOURCE
                     51
                           ;
                                 EXAMPLE FORTRAN CALL
                     52
                     53
                           ;
                                       SUBROUTINE TEST
                     54
                                       DIMENSION XARRAY(5), ZARRAY(3), YARRAY(15)
                     55
                           į
                                       INTEGER XINDEX, ZINDEX, NXPTS, NZPTS
                     56
                                       COMMON /STUFF/XINDEX,ZINDEX,NXPTS,NZPTS,ZARRAY, XARRAY,YARRAY
                                       DATA XINDEX,ZINDEX,NXPTS,NZPTS/0,0,5,3/
                     57
                           ;
                     58
                           ţ
                                       DATA ZARRAY/0.0,10.0,20.0/
                     59
                           ;
                                       DATA XARRAY/0.0,2.0,4.0,6.0,8.0/
                     60
                           ;
                                       DATA YARRAY/0.0,2.,4.,5.,8.,10.,12.,14.,16.,18.,20.,22.,24.,26.,28./
                     61
                           ;
                                         X = 2.5
                     62
                           ;
                                         Z=12.0
                     63
                                         Y=RFUN2L(X,Z,XINDEX)
                     64
                                         PRINT *, Y
                     65
                                       END
                     66
                     67
                           $#$#$#$#<del>$$$$$$$$$$$$$$$$$$$$$$$$$$$</del>
                     68
                     69
                     70
                                 NAME
                                       RFUN2L
                     71
                           CGROUP GROUP
                                       CODE
                     72
                           DGROUP GROUP
                                       DATA
                     73
                                 ASSUME CS:CGROUP, DS:DGROUP, ES:DGROUP
                     74
                                 PUBLIC RFUN2L
                     75
                          76
                     77
                     78
                                 STACK STRUCTURE
                     79
                     80
                          PARAMS STRUC
                     81
0000
                     82
                          OLD_ES DW
                                       ?
                                             SAVED ES REGISTER
0002
                     83
                           GLD_BP DW
                                       ?
                                             :SAVED BP REGISTER
0004
                     84
                          OLD_DS DW
                                       ?
                                             ;SAVED DS REGISTER
0006
                     85
                          RETURN DD
                                       ?
                                             ; RETURN ADDRESS (OFFSET ON TOP, THEN SEGMENT)
000A
                          DADDR DD
                     88
                                       ?
                                             ;ADDRESS OF DATA STRUCTURE (OFFSET, THEN SEGMENT)
000E
                     87
                          ZΥ
                                 DD
                                       ?
                                             ; ADDRESS OF ZV (OFFSET, THEN SEGMENT)
0012
                     88
                          XΛ
                                 DD
                                             ; ADDRESS OF XV (OFFSET, THEN SEGMENT)
                     89
                     90
                          PARAMS EMDS
                     91
                     92
                          93
                          94
                     95
                          DATA
                                SEGMENT PUBLIC 'DATA'
                     96
0000 ????
                     97
                          STWSAVE DW
                                       ?
                     98
                     99
                          DATA
                                ENDS
                    100
                    101
                          102
                          103
                    104
                          CODE
                                SEGMENT PUBLIC 'CODE'
                    105
```

LOC OBJ		LINE	SOURCE				
0000		106	FARCODE	PROC	FAR		
0000		107	HUNGODE	11100	1141		
		108					
		109					
0000		110	RFUN2L:				
0000 1E		111		PUSH	DS	SAVE F	REGISTERS
0001 55		112		PUSH	EP		
0002 06		113		PUSH	ES CD		
0003 8BEC		114		VOM VOM	BP, SP AX, DATA		
0005 B9	R	115 116		MOV	ES, AX		
0008 8EC0 000A C55E12		117		LDS	BX, [BP].XV	GET X	V ADDRESS AND SEGMENT
000H CSSE12		117		FLD	DWORD PTR [BX]		V ON TOP OF 8087 STACK
0010 C55E0E		119		LDS	BX, [BP].ZV		V ADDRESS AND SEGMENT
0013 9BD907		120		FLD	DWORD PTR [BX]	:PUT Z	v on top of 9037 stack
0016 C55E0A		121		LDS	BX, [BP].DADDR	#GET S	TART ADDRESS OF DATA STRUCTURE
0019 SBD3		122		MOV	DX, BX	SAVE	BX
001B 8B0F		123		MOV	CX, [BX].XPTR	;CX =	XPTR
CO1D 8B6F04		124		VOH	BP, [BX].NXPTS	;BP =	
0020 8B7F02		125		MOV	DI, [BX].ZPTR	;DI =	ZPTR
0023 D1E7		126		SHL	DI, 1		
0025 B1E7		127		SHL	DI, 1		BYTE INDEX = ELEMENT INDEX * 4
0027 887706		128		MOV	SI, [BX].NZPTS	;SI =	NZP15
002A D1E6		129		SHL	SI, i	•07 -	NO. OF BYTES = NZPTS * 4
002C D1E6		130		SHL	SI, 1	151 =	NO. OF BITES - N2FTS * 4
002E		131 132	GET_Z_I	MOEY:			
002E 83C308		133	UL:_4_1	ADD	BX, S	;RX =	START OF Z ARRAY
002E 63C3V6 0031 9BD811		134 +2		FCOH	DWORD PTR [BX + DI]	;	COMPARE ARRAY ELEMENT WITH ZV
0034 9B26DD3E0000	R	135 +1		FSTSW	ES:STWSAVE	Ť	
003A 9B		135 +1		FWAIT			
003B 25A10000	R	137 +1		HOV	AX. ES:STHSAVE		
003F 250041		138 +1		and	AX, 4100H	;	MASK-IN COMPARISON BITS
0042 3B0001		139 +2		CMP	AX, 0100H		THE THE STATE OF STAT
0045 7434		140 +2		JE	LEFT_OF_Z_START		; EXIT IF ZV < ARRAY ELE
		444 .0	MENT				
		141 +2 142				: IF 7	> DATA(ORIGINAL INDEX)
0047		143	RIGHT (DF_Z_STAF	χτ:		THEN BEGIN ***
0047 83C704		144	(120)11-	ADD	DI, 4		NDEX = INDEX + 4
004A 3BFE		145	RZS1:	CHP	DI, SI		HILE (INDEX <= NO. OF BYTES)
004C 741B		146		JE	RZS2	;	
		147					*** DO BEGIN ***
004E 9BD311		148 +2		FCOM	DWORD PTR [BX + DI]	;	COMPARE ARRAY ELEMENT WITH IV
0051 9B26DD3E0000	R	149 +1		FSTSW	ES:STWSAVE		
0057 9B		150 +1		FWAIT			
0058 26A10000	R	151 +1		VOK	AX, ES:STHSAVE		MACH IN COMPADICON DITC
005C 250041		152 +1		AND	AX, 4100H	;	MASK-IN COMPARISON BITS
005F 3D0001		153 +2 154 +2		CMP JE	AX, 0100H RZS2	;	EXIT IF ZV < ARRAY ELEMENT
0062 74 05		155 +2		UL.	NEGE .	,	
0054 830704		156		ADD	DI, 4	;	INDEX = INDEX + 4
0067 EBE1		157		ÚMÞ	RZS1	;	*** END (WHILE) ***
0069 8BC6		158	RZS2:	MOV	AX, SI	;	
006B 2B0400		159		SUB	AX, 4	;	

LOC OBJ	LINE	SOURCE		
	222			
006E 83EF04	160	SUB	DI, 4	;
0071 3BF8	161	CHP	DI, AX	; IF INDEX O NO. OF BYTES
0073 7542	162	JNE	FOUND_LOW_Z_INDEX	; THEN FOUND_LOW_INDEX
0075 9BDDD8	163	FSTP	ST(0)	; ELSE OUTLOF_RANGE
0078 EB4D90	164	JMP	GET_X_INDEX	;
	165			:*** END (IF) ***
	166			; IF Z < DATA(CRIGINAL INDEX)
007B	167	LEFT_OF_Z_ST/	ART:	:*** THEN BEGIN ***
007B 83EF04	168	SUB	DI, 4	; INDEX = INDEX - 4
007E 83FF00	169	LZS1: CMP	DI, 0	; WHILE (INDEX > 0)
0081 7020	170	U231. UL	LZS2	• KAILE (TABEA > 0)
AA01 10TA	171	UL.	1202	e ver Po DECTN ver
0083 9BD811	172 +2	FOOM	Discount of the core	; *** DO BEGIN ***
0086 9B26DD3E0000 R	172 +2		DWORD PTR [BX + DI]	: COMPARE ARRAY ELEMENT WITH ZV
008C 9B				
	174 +1	FHAIT		
008D 26A10000 R	175 +1		AX, ES:STHSAVE	
0091 250041	176 +1	AND	AX, 4100H	: MASK-IN COMPARISON BITS
0094 3D0000	177 +2		AX, 0000H	; EXIT IF ZV > ARRAY ELEMENT
0097 740A	178 +2		LZS2	
0099 3D0040	179 ÷2	CMP	AX, 4000H	
009C 7405	180 +2	JE	LZS2	; EXIT IF ZV = ARRAY ELEMENT
009E 83EF04	181	SUB	DI, 4	; INDEX = INDEX - 4
OOA1 EBDB	182	q _H p	LZS1	; *** END (WHILE) ***
00A3 83FF00	183	LZS2: CMP	PI, 0	; IF INDEX ⇔ 0
00A6 7D0F	184	JGE	FOUND_LOW_Z_INDEX	THEN FOUND_LOW_INDEX
00A8 83C704	185	ADD	DI, 4	: ELSE OUT_OF_RANGE
OOAB 9BDDDS	186	FSTP	ST(0)	:
OOAE 9BD9EE	187	FLDZ	0.107	REPLACE TOP WITH O
00B1 9BD9C9	188	FXCH	ST(1)	; PUT XV ON TOP OF 8087 ST.
00B4 EB1190	189	JHP	GET_X_INDEX	, 191 WA OU 101 OL 0007 21.
VV2 (2211) V	190	Oth	CETENETHEEX	; ;*** END(IF) ***
	191			YANA CHOILL AAA
00B7	192	FOUND_LOW_Z_1	FIDEA.	
00B7 9BD901	193			- STORE TI ON STAGE TOR
OOBA 9BDCE9	194	FLD	DMCRD PTR [BX+DI]	STORE ZL ON STACK TOP
00BD 9BD86904		FSUB	ST(1), ST	(ZV-ZL) 2ND FROM TOP
	195	FSUBF	N DWORD PTR [BX+DI+4]	; (ZH-ZL) ON STACK TOP
00C1 9BBEF9	196	FDIV		:(ZV-ZL)/(ZH-ZL) LEFT ON STACK
00C4 9PD9C9	197	FXCH	ST(1)	PUT XV ON TOP OF 8087 STACK
6607	198			
0007	199	GET_X_INDEX:	-	
00C7 03BE	200	ADD	BX, SI	;BX = START OF XARRAY
00C9 87F9	201	XCHG	DI, CX	:DI = XPTR : CX = LOW_Z_INDEX
00CB D1E7	202	SHL	DI, 1	
00CB D1E7	203	SHL	DI, 1	DI = BYTE INDEX = XPTR * 4
00CF 87F5	204	XCHG	SI, BP	SI = NXPTS
00D1 D1E5	205	SHL	SI, 1	•
00D3 D1E6	205	SHL	SI, i	:SI = NO. OF X BYTES = NXPTS * 4
00D5 9BD811	207 +2	FCOM	DWORD PTR [BX + DI]	COMPARE ARRAY ELEMENT WITH XV
00D8 9B26DD3E0000 R	208 +1	FSTSI		
00DE 9B	209 +1	FWAIT	•	
00DF 26A10000 R	210 +1	VOM	AX: ES:STWSAVE	
00E3 250041	211 +1	AND	AX, 4100H	; MASK-IN COMPARISON BITS
00E6 3D0001	212 +2	CMP	AX, 0100H	The second section of the second section secti
00E9 7434	213 +2	JE	LEFT_OF_X_START	; EXIT IF XV < ARRAY ELE
		MENT		, LAIT II AV CHINH! ELE

LOC OBJ	LINE	SOURCE		
	214 +2			
	215			; IF X > DATA(ORIGINAL INDEX)
00EB	216	RIGHT_OF_X_ST		;*** THEN BEGIN ***
00EB 83C704	217	ADD	DI, 4	; INDEX = INDEX + 4
OOEE 3BFE	218	RXS1: CMP	DI, SI	; WHILE (INDEX <= NO. OF BYTES)
00F0 741B	219	JE	RXS2	1
	220			; *** DO BEGIN ***
00F2 9BD911	221 +2	FCOM	DWORD PTR [BX + DI]	; COMPARE ARRAY ELEMENT WITH XV
00F5 9B25DB3E0000 R		FSTSW	ES: STWSAVE	
00FB 9B	223 +1	FWAIT	AU MA ATUATUM	
00FC 26A10000 R		VON	AX, ES:STWSAVE	WARE THE SOUDANTSON STYS
0100 250041	225 +1	AND	AX, 4100H	; MASK-IN COMPARISON BITS
0103 3B0001	226 +2	CMP	AX, 0100H	. PUTT TE UN Z ARGAU ELEMENT
0105 7405	227 +2 228 +2	JE	RXS2	; EXIT IF XV < ARRAY ELEMENT
0108 830704	229	ADD	DI, 4	; INDEX = INDEX + 4
0108 EBE1	230	JMP	RXS1	; *** END (WHILE) ***
010D 8BC6	231	RXS2: MOV	AX, SI	•
010F 2D0400	232	SUB	AX, 4	· •
0112 83EF04	233	SUB	DI, 4	;
0115 3BF8	234	CMP	DI, AX	: IF INDEX <> NO. OF BYTES
0117 753F	235	JNE	FOUND_LGW_X_INDEX	; THEN FOUND_LOW_INDEX
0117 733F 0119 9BDDB8	236	FSTP	ST(0)	; ELSE OUT_OF_RANGE
011C EB4790	237	JMP	GET_YL	;
Offo CD1110	233	O. a.	*212.2	:*** END (IF) ***
	239			; IF X < DATA(ORIGINAL INDEX)
011F	240	LEFT_OF_X_STA	RT:	;*** THEN BEGIN ***
011F 83EF04	241	SUB	DI, 4	; INDEX = INDEX - 4
0122 83FF00	242	LXS1: CMP	BI, 0	; WHILE (INDEX > 0)
0125 7020	243	JL	LXS2	;
V115 / V10	244			; *** DO BEGIN ***
0127 9BD311	245 +2	FCOM	DWORD PTR [BX + DI]	; COMPARE ARRAY ELEMENT WITH XV
012A 9B26DD3E0000 F		FSTSk	ES: STWSAVE	
0130 9B	247 +1	FWAIT	,	
0131 26A10000 F	248 +1	HOV	AX, ES:STWSAVE	
0135 250041	249 +1	AND	AX, 4100H	; MASK-IN COMPARISON BITS
0138 3D0000	250 +2	CMP	AX, 0000H	; EXIT IF XV > ARRAY ELEMENT
013B 740A	251 +2	JE	LXS2	
013D 3D0040	252 +2	CMP	AX, 4000H	
0140 7405	253 +2	JE	LXS2	; EXIT IF XV = ARRAY ELEMENT
0142 83EF04	254	SUB	DI, 4	: INDEX = INDEX - 4
0145 EBDB	255	J H P	LXS1	; *** END (WHILE) ***
0147 83FF00	256	LXS2: CMP	DI, 0	; IF INDEX \diamondsuit 0
014A 7D0C	257	J∂E	FOUND_LOW_X_INDEX	; THEN FOUND_LOW_INDEX
0140 830704	258	ADD	DI, 4	; ELSE OUT_OF_RANGE
014F 9BDDD3	259	FSTP	ST(0)	i
0152 9BD9EE	260	FLDZ		REPLACE TOP WITH 0
0155 EB0E90	261	JMP	GET_YL	;
	262			;*** END(IF) ***
0158	263 26 4	FOUND_LOW_X_	INDEY:	
	26 4 265	FLD	DWORD PTR [BX+DI]	STORE XL ON STACK TOP
0158 9BD901 015B 9BDCE9	265 266	FSUB		; (XV-XL) ON STACK BOTTOM
015E 9BD85904	267	FSUBI		;(XH-XL) ON STACK TOP
0162 9BDEF9	267 268	FDIV		;(XV-XL)/(XH-XL) LEFT ON STACK
VIGZ FDUCEFF	200	1 114		THE REPUBLIC REF EET OR OTHER

LOC	GBJ	LINE	SOURCE			
		269				
0165		270	GET_YL:			
	03DE	271	OLILIE.	ADD	BX, SI	:BX = START OF YARRAY
	8BC6	272				AX = NO. OF X BYTES
				VOK	AX, SI	SRX = NU. UF X BYIES
	D1E8	273		SHR	AX, 1	
	D1E8	274		SHR	AX, 1	AX = NXPTS = NO. OF X BYTES / 4
016D	F6E1	275		MUL	CL	AX = YA RON OFFSET
		276				THE AVOVE INSTR. LIMITS LENGTH OF ZARRAY
		277				;TO 255 ELEMENTS
016F	03D8	278		ADD	BX, AX	:BX = YA RON PTR = NXPTS * ZPTR
0171	03DF	279		ADD	BX, DI	;BX = YA POINTER
0173	9BD907	280		FLD	DWORD PTR [BX]	STORE YA ON ST TOP
	8BC7	281		MOV	AX, DI	
	050400	282		ADD	AX, 4	
	3BC6	283		CMP	AX, SI	
	7410	284		JE	GET_YH	FEXIT IF XV PAST BOUNDRY
	9BD9C9	285		FXCH	ST(1)	;(XV-XL)/(XH-XL) ON TOP, THEN YA
	98094704	286		FLD	DWORD PTR [BX + 4]	STORE YE ON ST. TOP
	9BD8E2			-		
		287		FSUB	ST, ST(2)	(YB-YA) ON ST. TOP
	99D809	288		FMUL	ST, ST(1)	;((XV-XL)/(XH-XL))(YB-YA) ON ST TOP
0180	9BDEC2	289		FADDP	ST(2), ST	YL 2ND FROM TOP OF STACK
24.05		290	555 Lat.			
018F		291	GET_YH:			
	8BC1	292		MOV	AX, CX	; IF (Z_POINTER TOO BIG)
0191	050400	293		ADD	AX, 4	;*** DO BEGIN ***
0194	3BC5	294		CMP	AX, BP	
0196	750F	295		JME	GYH1	
0198	8BC7	296		VOM	AX, DI	; IF (X_POINTER TOO BIG)
019A	050400	297		ADD	AX, 4	
019B	3BC6	293		CMP	AX, SI	
	742A	299		JΈ	FOUND_YV	: THEN EXIT
	9BDDD8	300		FSTP	ST	; ELSE DISCARD JUNK
	EB2590	301		JMP	FOUND_YV	; AND THEN EXIT;
		302				;*** END (IF) ***
0167	' 03DE	303	GYH1:	ADD	BX, SI	:BX = YC POINTER
	9BD907	304	• • • • • • • • • • • • • • • • • • • •	FLD	DWORD PTR [BX]	STORE YOUN STACK TOP
	8BC7	305		MOV	AX, DI	TOTOLE TO DIV OTHER TO
	050400	306		ADD	AX, 4	
	3BC6	307		CMP	AX, SI	
	740D	308		JE	GET_YV	;EXIT IF XV PAST BOUNDRY
	9PD94704	309		FLD	DWORD PTR [BX + 4]	STORE YD ON STACK TOP
	9BD3E1	310		FSUB	ST, ST(1)	;YD-YC ON ST TOP
	9BDECA	311		FMULP	ST(2), ST	(((XV-XL)/(XH-XL))(YD-YC) 2ND FROM TOP
01BF	9BDEC1	312		FADD		;YH ON TOP OF 8087
		313	: : :			
0102		314	GET_YV:	====		;YH TOP, THEN YL, THEN SLOPE
	9BD8E1	315		FSUB	ST, ST(1)	;YH-YL ON ST. TOP
	9BDECA	316		FMULP	ST(2), ST	;((ZV-ZL)/(ZH-ZL))(YH-YL) 2ND FROM TOP
0108	9BDEC1	317		FADD		YV ON ST. TOP
		318				
0108		319	FOUND_Y			
01CB		320	SAVE_NE	WLPOINTE		
	8BDA	321		MOV	BX, DX	
	D1EF	322		SHR	DI, 1	
010F	DIEF	323		SHR	DI, 1	

LOC	OBJ CBJ	LINE	SOURCE				
01D1	893F	324		MOV	[BX].XPTR,	DI	;SAVE XPTR
0103	D1E9	325		SHR	CX, 1		
0105	D1E9	326		SHR	CX, 1		
01D7	894F02	327		MOV	[BX].ZPTR,	CX	SAVE ZPTR
		328					
01DA		329	EPILOGUE	:			
01DA	07	330		POP	ES		
OIDB	5D	331		P0P	₽P		RESTORE REGISTERS
01DC	1F	332		POP	DS		
OIDD	CA0C00	333		RET	12		
		334					
		335	FARCODE	ENDP			
		336					
	•	337	CODE	ENDS			
		338					
		339	; *****	******	********	**********	*******************
		340	*****	*****	******	********	*************************************
		341					
		342		END			

ASSEMBLY COMPLETE, NO ERRORS FOUND

SERIES-III 8086/87/88/186 MACRO ASSEMBLER V2.0 ASSEMBLY OF MODULE REUNSS OBJECT MODULE PLACED IN :F1:RFUN3S.OBJ ASSEMBLER INVOKED BY: ASM86.86 :F1:RFUN3S.SRC

LOC OBJ	LINE	SOURCE	
	i	;	********
	2	•	* ROUTINE MAME: RFUN3S *
	3	;	* DATE: SEP. 6, 1983 *
	4	•	* FROGRAMMER'S NAME: MICHAEL MACKIN *
	5	:	* PURPOSE: TO INTERPOLATE A SIMPLE *
	6	,	* FUNCTION FOR SMALL-MODEL *
	7	•	* PROGRAMS *
	8	•	* 1100/4710 *
	9	,	**************************************
	-	,	THIS SUBSCRIPTION SUPPLIES THE PATHWAYER HALLE OF A PRINCIPLO ME. COMM. THE
	10	7	THIS SUBROUTINE OUTPUTS THE ESTIMATED VALUE OF A FUNCTION YV = F(XV, ZV)
	11	i	BY INTERPOLATING FROM KNOWN SOLUTION POINTS, USING THE RELATIONS
	12	;	XF = ((ZV-ZL)/(ZH-ZL))(XB-XC) + XC
	13	;	XG = ((ZY-ZL)/(ZH-ZL))(XE-XD) + XD
	14	;	YF = ((ZV-ZL)/(ZH-ZL))(YB-YC) + YC
	15	;	YG = ((ZV-ZL)/(ZH-ZL))(YE-YD) + YD
	16	7	YV = ((XV-XF)/(XG-XF))(YG-YF) + YF
	17	;	
	18	;	INPUT REQUIREMENTS:
	19	3	 THE FOLLOWING ADDRESSES WILL BE PASSED TO THIS SUBROUTINE:
	20	;	A. ON TOP OF THE 8086 STACK:
	21	;	THE ADDRESS (OFFSET) OF A DATA AREA WITH THE FOLLOWING FORMAT
	22	ţ	
	23		DSTRUC STRUC
	24		##1116A A1116A
0000	25		XPTR DW ? ;X ARRAY INDEX
0002	26		***************************************
0004	27		
0005	28		
0005			
	29	;	ZARRAY DD NZPTS DUP(?) ;Z ARRAY (255 ELEHENTS MAXIMUM)
	30	;	XARRAY DD NXPTS DUP(?) ; X ARRAY
	31	;	YARRAY DD NYPTS DUP(?) ;Y ARRAY
	32		
~~~	33		DSTRUC ENDS
	34	5	
	35	;	B. ON THE 8087 STACK: ON TOP ZV, THEN XV
	36	;	
	37	;	<ol><li>IT IS REQUIRED THAT THE 8087 CHIP HAVE 5 EMPTY REGISTERS</li></ol>
	38	;	WHEN THIS INTERPOLATION ROUTINE IS CALLED.
	39	•	
	40	;	3) NOTE: NO TWO ADJACENT ELEMENTS OF XARRAY, YARRAY, OR ZARRAY
	41	;	MAY HAVE IDENTICAL ELEMENTS. THIS WILL RESULT IN DIVISION
	42	;	BY ZERO.
	43	;	and the terms of t
	44	;	OUTPUT EFFECTS:
	45	;	1) REGISTERS DESTROYED: AX, BX, CX, DX, DI, SI
	46	;	2) INTERPOLATED RESULT RETURNED ON TOP OF 8087 STACK
	47	;	3) XPTR LOCATION UPDATED TO INDEX X VALUE JUST PRECEDING XV.
	48	;	ZPTR LOCATION UPDATED TO INDEX Z VALUE JUST PRECEDING ZV.
	49	;	(TO SPEED FUTURE SEARCHES FOR X AND Z)
	50	,	et man / et et le Gallingting / Git A 1812 E/

```
LOC OBJ
                      LINE
                               SCURCE
                                     EXAMPLE PL/M CALL
                        51
                         52
                         53
                                     PLHPGM: DO;
                         54
                                             DECLARE DSTRUC STRUCTURE (
                         55
                               ;
                                                XINDEX INTEGER, ZINDEX INTEGER, NXPTS INTEGER, NZPTS INTEGER,
                         56
                                                ZARRAY(3) REAL, XARRAY(15) REAL, YARRAY(15) REAL) PUBLIC
                         57
                               ;
                         58
                                                INITIAL (
                               ;
                         59
                               ;
                                                  /* XINDEX */ 0,
                                                  /* ZINDEX */ 0,
                         60
                               ;
                                                  /* NXPTS */ 5,
                         61
                               ;
                                                  /* NZPTS */ 3,
                         62
                               ţ
                         63
                                                  /* ZARRAY */ 0.0, 10.0, 20.0,
                               ij
                                                  /* XARRAY */ 0.0, 2.0, 4.0, 6.0, 8.0,
                         64
                               ţ
                                                             1.0, 3.0, 5.0, 7.9, 9.0,
                         65
                               ï
                                                              1.5, 1.75, 1.8, 5.0, 10.0,
                         66
                               ï
                                                  /* YARRAY */ 0.0, 2.0, 4.0, 6.0, 8.0,
                         67
                               ;
                                                              10.0, 12.0, 14.0, 16.0, 18.0,
                         68
                                                              20.0, 22.0, 24.0, 26.0, 28.0 );
                         69
                               ;
                         70
                               ţ
                                             RFUNGS: PROCEDURE(X,Z,STRUC_ADDR) REAL EXTERNAL;
                         71
                         72
                                                    DECLARE X REAL;
                                                    DECLARE Z REAL;
                         73
                                                    DECLARE STRUC_ADDR POINTER:
                         74
                         75
                               ;
                                             END RFUN3S;
                         76
                               ÷
                                             R3TEST: PROCEDURE PUBLIC;
                         77
                                                    DECLARE (X,Z,Y) REAL;
                         78
                         79
                                                       X = 3.2
                               ,
                         90
                                                       Z = 12.0;
                                                       Y = RFUN3S(X, Z, @DSTRUC);
                         81
                                                       CALL PRINTOUT(Y);
                         82
                                                    END;
                         83
                                             END R3TEST:
                         84
                               i
                         85
                               ;
                         86
                                      END PLMPGH;
                         87
                               83
                         89
                         90
                                             RFUN3S
                         91
                                      MAME
                               CGROUP GROUP
                                             CODE
                         92
                         93
                               DGROUP GROUP DATA
                                       ASSUME CS:CGROUP, DS:DGROUP
                         94
                                      PUBLIC RFUNGS
                          95
                          96
                               97
                         98
                         99
                                      STACK STRUCTURE
                               ;
                         100
                         101
                               PARAMS STRUC
                         102
                                                     ;SAVED BP REGISTER
                                              ?
 0000
                         103
                                OLD_BP DW
                                              ?
                                                     RETURN ADDRESS
 0002
                         104
                                RETURN DH
                                DADDR DW
                         105
                                              ?
                                                     :ADDRESS OF DATA STRUCTURE
 0004
```

LOC OBJ	LINE	SOURCE		
	106 107 108	PARAMS ENDS		
	109	;* <del>*******</del>	**********	*******
	110	; **********	***********************	*********
	111	DATA CECH	THE DUDI TO ADATAA	
	112 113	DATA SEGM	ENT PUBLIC 'DATA'	
0000 ????	114	STUSAVE DW	?	
	115			
	116	DATA ENDS		
	117			
	118 119			**************************************
	120	3 00 00 00 00 00 00		**********
	121	CODE SEGM	ENT PUBLIC 'CODE'	
	122			
0000	123	NEARCODE	PROC NEAR	
	124 125			
	126			
0000	127	RFUN3S:		
	128			SAVE REGISTERS
0000 55	129	PUSH		
0001 8BEC 0003 8B5E04	130 131	VOM VOM	BP, SP BX, [8P].DADDR	GET START ADDRESS OF DATA STRUCTURE
0006 8BD3	132	VOM	DX, BX	SAVE BX
0008 8B2F	133	VOM	BP, [BX].XPTR	:BP = XPTR
000A 8B4F04	134	VCM	CX, [BX].NXPTS	CX = NXPTS
000D 8B7F02	135	VOM	DI, [BX].ZPTR	:DI = ZPTR
0010 D1E7 0012 D1E7	136 137	SHL SHL	DI, 1 DI, 1	;DI = BYTE INDEX = ELEMENT INDEX * 4
0014 8B7706	138	HOV	SI, [BX].NZPTS	SI = MZPTS
0017 D1E6	139	SHL	SI, 1	
0019 D1E6	140	SHL	SI, 1	SSI = NO. OF BYTES = NZPTS * 4
0010	141	GET_Z_INDEX:		
001B 001B 83C308	142 143	OET_Z_INDEX:	BX, 8	;BX = START OF Z ARRAY
001E 9BD811	144 +2			; COMPARE ARRAY ELEMENT WITH ZV
	R 145 +1	. FSTS	N STNSAVE	
0026 9B	146 +1			
0027 A10000 002A 250041	R 147 +1 148 +1		AX, STWSAVE AX, 4100H	; MASK-IN COMPARISON BITS
002D 3D0001	149 +2		AX, 0100H	) HON IN CONTINUOUS DITO
0030 7432	150 +2	? JE	LEFT_OF_Z_START	; EXIT IF ZV < ARRAY ELE
		HENT		
	151 +2	2		ATT TO BATA/ODIOTHAL TAREYS
0032	152 153	RIGHT_OF_Z_S	TART:	;IF Z > DATA(ORIGINAL INDEX) ;*** THEN BEGIN ***
0032 830704	154	ADD	DI, 4	; INDEX = INDEX + 4
0035 3BFE	155	RZS1: CMP	DI, SI	; WHILE (INDEX <= NO. OF BYTES)
0037 7419	156	JE	RZS2	i ver DO SECIN ver
0039 9BD811	157 158 +2	2 FCOF	DWORD PTR [BX + DI]	; *** DO BEGIN *** ; COMPARE ARRAY ELEMENT WITH ZV
	R 159 +1			. Secretaria construction deliberation (TATIL A.Y

LOC OBJ		LINE	SOURCE			
0041 9B		160 +1		FWAIT		
0042 A10000	R	161 +1		MOV	AX, STWSAVE	. WASH IN COMPARISON BITS
0045 250041		162 +1		AND	AX, 4100H	; MASK-IN COMPARISON BITS
0048 3D0001		163 +2		CMP	AX, 0100H	. FVIT IF 711 / ADDAM FI EMENT
004B 7405		164 +2		JΕ	RIS2	; EXIT IF ZV < ARRAY ELEMENT
**** *******		165 +2		ADD	DT 4	. INDEX - THREE : A
004D 83C704		166		ADD	DI, 4 RZS1	;
0050 EBE3		167	07001	JHP	AX, SI	· *** END (MUILE) ***
0052 8BC6 0054 2B0400		168	RZS2:	MOV SUB	AX, 4	) *
0054 280400 0057 83EF04		169 170		SUB	DI, 4	;
005A 3BF8		171		CHP	DI, AX	; IF INDEX $\bigcirc$ NO. OF BYTES
005C 7540		172		JNE	FOUND_LOW_Z_INDEX	THEN FOUND_LOW_INDEX
005E 9BDDD8		173		FSTP	ST(0)	; ELSE OUT_OF_RANGE
0061 EB4B90		174		JMP	GET_X_INDEX	•
0301 22 1270		175		0.11	2012/12/12/12/1	:*** END (IF) ***
		176				; IF Z < DATA(ORIGINAL INDEX)
0064		177	LEFT_09	Z_STAR	ī:	;*** THEN BEGIN ***
0064 83EF04		178		SUB	DI, 4	; INDEX = INDEX - 4
0067 83FF00		179	LZS1:	CMP	DI, 0	; WHILE (INDEX > 0)
006A 7C1E		180		JŁ	LZS2	•
		181				; *** DO BEGIN ***
006C 9BD311		182 +2		FCOM	DWORD PTR [BX + DI]	; COMPARE ARRAY ELEMENT WITH ZV
006F 9BDD3E0000	R	183 +1		FSTSH	STHEAVE	
CO74 9B		184 +1		FWAIT		
0075 A10000	R	185 +1		MOV	AX, STWSAVE	
0078 250041		186 +1		and	AX, 4100H	; MASK-IN COMPARISON BITS
007B 3D0000		187 +2		CMP	VX. 0000H	; EXIT IF ZV > ARRAY ELEMENT
007E 740A		188 +2		JΕ	LZS2	
0080 3D0040		189 +2		CMP	AX1 4000H	EVET TE THE ABOVE DIEMPHT
0083 7405		190 +2		JE	LZS2	; EXIT IF ZV = ARRAY ELEMENT
0085 83EF04		191		SUB	DI, 4	; INDEX = INDEX - 4
0088 EBDD		192 193	1.700	JHP CHP	LZS1 DI, O	; *** END (WHILE) *** ; IF INDEX <> 0
008A 83FF00 008D 7D0F		173	LZS2:	unr JGE	FOUND_LOW_Z_INDEX	; THEN FOUND_LOW_INDEX
008F 83C70 <b>4</b>		195		ADD	DI, 4	; ELSE OUT_OF_RANGE
0092 9BDDD3		176		FSTP	ST(0)	i
0095 9BD9EE		197		FLDZ	01107	REPLACE TOP WITH 0;
CO98 9BD9C9		198		FXCH	ST(1)	;
009B EB1190		199		JHP	GET_X_INDEX	;
		200				;*** END(IF) ***
		201				
009E		202	FOUND_	LOW_Z_IN	DEX:	
009E 9BD901		203		FLD	DWORD PTR [EX+DI]	STORE ZL ON STACK TOP
00A1 9BDCE9		204		FSUB	ST(1), ST	;(ZV-ZL) 2ND FROM TOP
00A4 9BB86904		205		FSUBR	EWORD PTR [BX+DI+4]	(ZH-ZL) ON STACK TOP
OOAS 9BDEF9		206		FDIV		;(ZV-ZL)/(ZH-ZL) LEFT ON STACK
00AB 9BD9C9		207		FXCH	ST(1)	PUT XV ON TOP OF 8087 STACK
AAAF		208	A== "	THEEY-		
00AE		209	GET_X_		PV 01	*DV - CTAST OF VARSAV
OOAE O3DE		210		ADD	BX, SI	BX = START OF XARRAY
00B0 8BC7		211		HOV No.11	AX, DI	AX = NO. OF Z BYTES  AY = YO BOW DEECET N OF ZBYTEC NUMBER
00B2 F6E1		212		MUL	CL BY AY	AX = XC ROW OFFSET=N. OF ZBYTES*NXPTS
00B4 03D8		213		ADD	BX, AX	;BX = XC ROW PTR
00B6 87FD		214		XCHG	DI, BP	;DI = XPTR : BP = LON_Z_INDEX

LOC	OBJ		LINE	SOURCE				
00BA	D1E7 D1E7		215 216		SHL SHL	DI, 1 DI, 1	:DI =	BYTE INDEX = XPTR * 4
	87F1		217		XCHG	SI, CX		: NXPTS : CX = NO. OF Z BYTES
	D1E6		218		SHL	SI, 1	;	THE TOTAL OF E BIJES
	D1E6		219		SHL	SI, 1	fSI =	NO. OF X BYTES = NXPTS * 4
	9BD901		220		FLD	DWORD PTR [BX + DI]		E XC ON TOP OF STACK
	8PC5 050400		221		MOV	AX, BP		
00C7			222		ADD	AX, 4		
	7410		223 224		CMP	AX, CX		
00CE			225		JE ADD	GX1 BX, DI	\$EXIT	IF Z TOO HIGH
	9BD900		226		FLD	DWORD PTR [BX + SI]	erno	E VD ON TOD OF OTAGIA
00D3	2BDF		227		SUB	BX, DI	10100	E XB ON TOP OF STACK
0005	9BD8E1		228		FSUB	ST, ST(1)	; XR-X	C ON ST. TOP
	9BDSCB		229		FMUL	ST, ST(3)		-ZL)/(ZH-ZL))(XB-XC) ON ST. TOP
	9BDEC1		230		FADD			N ST. TOP
	9BD8D1			GX1:	FCOM	ST(1)	į	COMPARE ARRAY ELEMENT WITH XV
	9BDD3E0000	R	232 +1		FSTS₩	STUSAVE		
00E6			233 +1		FWAIT			
	A10000 250041	R	234 +1		VOM	AX, STWSAVE		
	3D0000		235 +1 236 +2		AND	AX, 4100H	;	MASK-IN COMPARISON BITS
00F0			237 +2		CMP JE	AX, 0000H LEFT_OF_X_START	;	EXIT IF XV > ARRAY ELEMENT
			238 +2		O.C.	LEFT_UF_X_STMRT		
			239				• TE Y	> DATA(ORIGINAL INDEX)
00F2			240	RIGHT_0	DF_X_STAF	<b>?T:</b>		FHEN BEGIN ***
	830704		241		ADD	DI, 4		VDEX = INDEX + 4
00F5	<del>-</del>		242	RXS1:	CMP	DI, SI		HILE (INDEX <= NO. OF BYTES)
00F7	/438		243		JΕ	RXS2	;	
ONEO	9BD901		244		<b>5</b> 1.5		;	*** DO BEGIN ***
00FC			245 246		FLD	DHORD PTR [BX + DI]	;	STORE XD ON TOP OF STACK
	050400		247		MOV ADD	AX, 6P AX, 4	;	
0101			248		CHP	AX, CX	;	
0103			249		JΕ	RXS3	? •	EVIT IE 7 TOO HIGH
0105	03DF		250		ADD	BX, DI	;	EXIT IF Z TOO HIGH
	9BD900		251		FLD	DWORD PTR [BX + SI]	;	STORE XE ON TOP OF STACK
010A :			252		SUB	BX, DI	;	OF STATE OF TO, OF STREET
0100 9			253		FSUB	ST, ST(1)	;	XE-XD ON ST. TOP
0112 S	9BD8CC		254		FHUL	ST, ST(4)	;	((ZV-ZL)/(ZH-ZL))(XD-XE) ON ST. TOP
0115			255	DVC2.	FADD	OTIO	;	XG ON ST. TOP
	PBDD3E0000	R	256 +2 257 +1	RADD:	FCOM FSTSW	ST(2) STHSAVE	;	COMPARE ARRAY ELEMENT WITH XV
0110 9		**	258 +1		FWAIT	JAHOHAE		
011E A		R	259 +1		MOV	AX, STMSAVE		
0121 2	250041		260 +1		AND	AX, 4100H	;	MASK-IN COMPARISON BITS
0124 3			261 +2		CMP	AX, 0000H	;	EXIT IF XV > ARRAY ELEMENT
0127 7	740B		262 +2 263 +2		JΈ	RXS2		The state of the s
0129 9	BD9C9		264		FXCH	CT/41		VE 10
0120 9			265		FSTP	ST(1) ST(0)	;	XF = XG
012F 8			266		ADD	DI, 4	;	DISCARD OLD XF
0132 E			267		JMP	RXS1	;	INDEX = INDEX + 4 *** END (HHILE) ***
0134 9	BD9C9		268	RXS2:	FXCH	ST(1)	;	*** CMD /MUITER ***
0137 8	BC6		269		MOV	AV	;	

LOC OBJ		LINE	SOURCE				
0139 2D0400		270		SUB	AX, 4	;	
013C 83EF04		271		SUB	DI, 4	;	
013F 3BF8		272		CW5	DI, AX	; IF	INDEX <> NO. OF BYTES
0141 7565		273		JNE	FOUND_LOW_X_INDEX	;	THEN FOUND_LOW_INDEX
0143 9BDDD9		274		FSTP	ST(0)	;	ELSE OUT_OF_RANGE
0146 9BDDD8		275		FSTP	ST(0)	5	DISCARD XG, AND XV
0149 EB7290		276		JWD	GET_ANSWER	;	**************************************
		277					ND (IF) ***
		278		- v ozasi	•-		< DATA(ORIGINAL INDEX)
0140		279	LEF I_U	-XLSTAR1			THEN BEGIN *** TNDEX = INDEX - 4
014C 83EF04		280		SUB	DI, 4		
014F 83FF00		281	LXS1:	CMP	DI, 0		HILE (INDEX > 0)
0152 7040		282		JL	LXS2	;	*** DO EEGIN ***
2454 005004		283		r) h	DWORD PTR [BX + DI]	;	STORE XD ON TOP OF STACK
0154 9BD901		284		FLD MOV	AX, EP	, ;	STORE AD CH TO OF STREET
0157 8905		285 286		ADD	AX, 4	•	
0159 050400		200 287		CMP	AX, CX	•	
015C 3BC1 015E 7410		288		JE	LXS3	•	EXIT IF Z TOO HIGH
0160 03DF		200 289		ADD	BX, DI	,	EXT II E 700 M201
0160 03Bi		290		FLD	DWORD PTR [BX + SI]	;	STORE XE ON TOP OF STACK
0152 7BB700 0155 2BBF		270		SUB	BX, DI	;	
0167 9BD3E1		292		FSUB	ST, ST(1)	;	XB-XC ON ST. TOP
016A 9BD8CC		293		FMUL	ST, ST(4)	;	((ZV-ZL)/(ZH-ZL))(XB-XC) ON ST. TOP
016D 9BDEC1		294		FADD		;	XG ON ST. TOP
0170 9BD8B2		295 +2	LXS3:	FCOH	ST(2)	;	COMPARE ARRAY ELEMENT WITH XV
0173 9BDD3E0000	R	296 +1		FSTSW	STNSAVE		
0178 9B		297 +1		FWAIT			
0179 A10000	R	298 +1		HOV	AX, STWSAVE		
0170 250041		299 +1		AMD	AX, 4100H	;	MASK-IN COMPARISON BITS
017F 3D0001		300 +2		CMP	AX, 0100H		
0182 7410		301 +2		JΕ	LXS2	;	EXIT IF XV < ARRAY ELEMENT
0184 3D0040		302 +2		CMP	AX, 4000H		
0187 740B		303 +2		JE	LXS2	;	EXIT IF XV = ARRAY ELEMENT
0189 9BD9C9		304		FXCH	ST(1)	i	XF=XG
018C 9BDDD8		305		FSTP	ST(0)	•	DISCARD OLD XF
018F 83EF04		306		SUB	DI, 4	i	INDEX = INDEX - 4
0192 EBBB		307		JMP	LXSI	, , ,	*** END (WHILE) ***
0194 83FF00		308	LXS2:	CMP	DI, O		F INDEX <> 0 THEN FOUND_LON_INDEX
0197 7D0F		309		JGE	FOUND_LOW_X_INDEX	;	ELSE OUT_OF_RANGE
0199 830704		310		ADD	DI, 4	;	ELDE USTLOFINANCE
019C 9BDDD8		311		FSTP FSTP	ST(0) ST(0)	•	DISCARD XC
019F 9BDDD8 01A2 9BD9EE		312 313		FLDZ	31107	;	REPLACE TOP WITH O
		314		JMP	GET_ANSWER	•	The grow to warm o
01A5 EB1690		315		UTIF	OC I LANGUEN	: ***	END(IF) ***
		316				,	Life (2)
01A8		317	FOLIND	AI_X_HOJ_	MFX:	;XF.	XG, XV, ZSLOPE ON ST.
O1A8 9BDCEA		318	. 001100	FSUB	ST(2), ST		XG, (XV-XF), ZSLOPE
OIAB 9BDEE9		319		FSUB			-XF), (XV-XF), ZSLOPE
01AE 9BDEF9		320		FDIV			/-XF)/(XG-XF)), ZSLOPE
01B1 8BC5		321		VOM	AX, BP	;IF 7	NOT TOO HIGH
01B3 050400		322		ADD	AX, 4		
01B6 3BC1		323		CH#P	. AX, CX		
01B9 7403		324		JΕ	GET_ANSWER		

LOC	0BJ	LIME	SOURCE			
01.04	9BD9C9	325		FXCH	ST(1)	. TUTH /711 71 \/ //711 71 \ ON OT TOO
VIDA	700707	325 326		FAUN	31(1)	; THEN (ZV-ZL)/(ZH-ZL) ON ST. TOP
01BD		327	GET_ANSH	ER:		
	8866	328		MOV	AX, SI	:AX = NO. OF X BYTES
	D1E8	329		SHR	AX, 1	THA - NO. OF A DITES
	D1E8	330		SHR	AX, 1	IAV - NYDIC - NO OF V DVIEC / A
V101	BIEG .	331		опп	HA1 I	TAX = NXPTS = NO. OF X BYTES / 4
0102	F6E1	332		MUL	Ct	CX = NO OF Z BYTES
0163	1021			FIUL	CL	;AX = YC ROW OFFSET
		333				THE AVOVE INSTR. LIMITS LENGTH OF ZARRAY
A40E	. 0000	334		400	TOV AV	TO 255 ELEMENTS
	03D8	335		ADD	BX, AX	BX = YC ROW PTR = NXPTS * ZPTR
	03DF	336		ADD	BX, DI	;BX = YC POINTER
	9BD907	337		FLD	DWORD PTR [BX]	STORE YC ON ST TOP
	8BC5	338		MOV	AX, BP	
	050400	339		ADD	AX, 4	
	3BC1	340		CMP	AX, CX	
0103	7510	341		JNE	Z_NOT_HIGH	
		342				
01D5		343	Z_TOO_HI			; IF Z OUT OF RANGE
	8BC7	344		MOV	AX, DI	;*** DO BEGIN ***
	050400	345		ADD	AX, 4	; IF X OUT OF RANGE
	3RC6	346		CMP	AX, SI	
	7444	347		JE	FOUND_YV	; THEN EXIT
01PE		348	X_VALID:			; (Z TOO HIGH BUT X IS IN RANGE)
	9BD94704	349		FLD	DWORD PTR [BX + 4]	; STORE XD
01E2	EB3290	350		JMP	GET_YV	;*** END (IF) ***
		351				
01E5		352	Z_NOT_HI	GH:		;IF Z VALID
	8207	353		MOV	AX, DI	;*** DO PEGIN ***
	050400	354		ADD	AX, 4	; IF X OUT OF RANGE
	3BC6	355		CMP	AX, SI	;
	7423	356		JΈ	Z_VALID	; THEN EXIT
01EE		357	POTH_VAL	ID:		; ELSE (BOTH XV AND ZV ARE IN RANGE)
	9BD9C9	358		FXCH	ST(1)	;(ZV-ZL)/(ZH-ZL) ON TOP, THEN YC
01F1	9BD900	359		FLD	DWORD PTR [BX + SI]	STORE YB ON ST. TOP
01F4	9BD8E2	360		FSUB	ST, ST(2)	;(YB-YC) ON ST. TOP
01F7	9BD8C9	361		FHUL	ST, ST(1)	;((ZV-ZL)/(ZH-ZL))(YB-YC) ON ST TOP
	9BDEC2	362		FADDP	ST(2), ST	;YF 2ND FROM TOP OF STACK
01FD	9BD94704	363		FLD	DWORD PTR [BX + 4]	STORE YD ON STACK TOP
0201	9BD94004	364			DWORD PTR [BX + SI + 4]	
0205	9BD8E1	365		FSUB	ST, ST(1)	;YE-YD ON ST TOP
0208	9BDECA	366		FMULP	ST(2), ST	;((ZV-ZL)/(ZH-ZL))(YE-YD) 2ND FROM TOP
020B	9BDEC1	367		FADD		: YG ON TOP OF 8087
020E	EB0690	368		JMP	GET_YV	;*** END (IF) ***
		369				
0211		370	Z_VALID:			:(X OUT OF RANGE BUT Z VALID)
0211	03DE	371		add	BX, SI	
0213	9BD907	372		FLD	DWORD PTR [BX]	
		373				
0216		374	GET_YV:			;YG TOP, THEN YF, THEN SLOPE
0216	9BD8E1	375		FSUB	ST, ST(1)	;YG-YF ON ST. TOP
0219	9BDECA	376		FMULP	ST(2), ST	;((XV-XF)/(XG-XF))(YG-YF) 2ND FROM TOP
0210	9BDEC1	377		FADD		:YV ON ST. TOP
021F	EB0190	378		JHP	FOUND_YV	
		379				

LOC	OBJ	LINE	SOURCE			
		390				
0222		381	FOUND_	YV:		
0222		382	SAVE_N	EN_POINT	ERS:	
0222	8BDA	383		VOM	BX, DX	RESTORE BY TO START OF DATA STRUC.
0224	DIEF	384		SHR	DI, 1	
0226		335		SHR	DI, 1	
0228	893F	386		MOV	[BX].XPTR, DI	SAVE XPTR
	D1ED	387		SHR	BP, 1	
0220	DIED	388		SHR	EP, 1	
022E	896F02	339		VOM	[BX].ZPTR, BP	;SAVE ZPTR
		390				
0231		391	EPILOG	UE:		
0231		392		POP	BP	RESTORE REGISTERS
0232	C20100	393		RET	i	
		394				
		395	NEARCO	DΕ	ENDP	
		396				
		397	CODE	ENDS		
		398				
		399	; *****	*****	*** <del>**************</del>	***********
		400	\$ #####	*****	** <del>*************</del> **	*********************
		401				
		402		END		

ASSEMBLY COMPLETE, NO ERRORS FOUND

SERIES-III 8086/87/88/186 MACRO ASSEMBLER V2.0 ASSEMBLY OF MODULE REUNBL OBJECT MODULE PLACED IN :F1:RFUNGL.OBJ ASSEMBLER INVOKED BY: ASM86.86 :F1:RFUN3L.SRC

LOC OBJ	LINE	SOURCE	
	1	;	**********
	2	;	* ROUTINE NAME: RFUNGL *
	3	;	* DATE: SEP. 8, 1983 *
	4	;	* PROGRAMMER'S MAME: MICHAEL MACKIN *
	5	;	* PURPOSE: TO INTERPOLATE A SIMPLE *
	6	;	* FUNCTION FOR LARGE-MODEL *
	7	2	* PROGRAMS *
	8	;	** <del>**************************</del>
	9	;	
	10	7	THIS SUBROUTINE OUTPUTS THE ESTIMATED VALUE OF A FUNCTION $YV = F(XV, ZV)$
	11	;	BY INTERPOLATING FROM KNOWN SOLUTION POINTS, USING THE RELATIONS
	12	;	XF = ((ZV-ZL)/(ZH-ZL))(XB-XC) + XC
	13	;	XG = ((ZV-ZL)/(ZH-ZL))(XE-XD) + XD
	14	3	YF = ((ZV-ZL)/(ZH-ZL))(YB-YC) + YC
	15	;	YG = ((ZV-ZL)/(ZH-ZL))(YE-YD) + YD
	16	;	YV = ((XV-XF)/(XG-XF))(YG-YF) + YF
	17	;	TUBLIT DESCRIPTION
	18	;	INPUT REQUIREMENTS:
	19	;	1) THE FOLLOWING ADDRESSES WILL BE PASSED TO THIS SUBROUTINE:
	20 21	;	A. ON TOP OF THE 8086 STACK:
	21	;	THE ADDRESS (OFFSET) OF A DATA AREA WITH THE FOLLOWING
	23	;	FORMAT
	23 24	,	DSTRUC STRUC
	25 25		DOINGE STRUC
0000	26		XPTR DW ? 3X ARRAY INDEX
0002	27		XPTR DW ? ;X ARRAY INDEX ZPTR DW ? ;Z ARRAY INDEX
0004	28		NXPTS DN ? INO. OF X BREAKPOINTS ON EACH Z LINE
0006	29		NZPTS DW ? INO. OF ELEMENTS IN Z ARRAY
	30	;	ZARRAY DD NZPTS DUP(?) ; Z ARRAY (255 ELEMENTS MAXIMUM)
	31	;	XARRAY DD NXPTS DUP(?) ; X ARRAY
	32	;	YARRAY DD NYPTS DUP(?) ; Y ARRAY
	33		
	34		DSTRUC ENDS
	35	;	
	36	;	B. ON THE 8087 STACK: ON TOP ZV, THEN XV
	37	;	
	38	i	2) IT IS REQUIRED THAT THE 8087 CHIP HAVE 5 EMPTY REGISTERS
	39	;	WHEN THIS INTERPOLATION ROUTIME IS CALLED.
	40	;	
	41	;	3) NOTE: NO TWO ADJACENT ELEMENTS OF XARRAY, YARRAY, OR ZARRAY
	42	,	MAY HAVE IDENTICAL ELEMENTS. THIS WILL RESULT IN DIVISION BY ZERO.
	43 44	; •	OUTPUT EFFECTS:
	45	;	1) REGISTERS DESTROYED: AX, BX, CX, DX, DI, SI
	46	•	2) INTERPOLATED RESULT RETURNED ON TOP OF 8087 STACK
	47	;	3) XPTR LOCATION UPDATED TO INDEX X VALUE JUST PRECEDING XV.
	48	;	ZPTR LOCATION OPDATED TO INDEX Z VALUE JUST PRECEDING ZV.
	49	•	(TO SPEED FUTURE SEARCHES FOR X AND Z)
	50	;	The state of the s

2

```
SOURCE
LOC OBJ
                   LINE
                     51
                          ;
                                EXAMPLE FORTRAN CALL
                     52
                          ;
                     53
                                       SUBROUTINE F3CALL
                          ;
                                       DIMENSION XARRAY(15), ZARRAY(3), YARRAY(15)
                     54
                                       INTEGER XINDEX, ZINDEX, NXPTS, NZPTS
                     55
                                       COMMON /STUFF/XINDEX,ZINDEX,NXPTS,NZPTS,ZARRAY, XARRAY,YARRAY
                     56
                     57
                                       DATA XINDEX, ZINDEX, NXPTS, NZPTS/0,0,5,3/
                     53
                                       DATA ZARRAY/0.0,10.0,20.0/
                                       DATA XARRAY/0.0,2.0,4.0,6.0,8.0,1.,3.,5.,7.9,9.,1.5,1.75,1.8,5.,10./
                     59
                                       DATA YARRAY/0.0,2.,4.,6.,8.,10.,12.,14.,16.,18.,20.,22.,24.,26.,28./
                     60
                                         X=9.1
                     61
                                         Z=11.3
                     62
                          ;
                                         Y=RFUN3L(X,Z,XINDEX)
                     63
                          ÷
                                         FRINT *, Y
                     64
                          ;
                                       END
                     65
                          ;
                     66
                           ;<del>************************</del>
                     67
                     68
                     69
                     70
                                 NAME
                                       RFUN3L
                           CGROUP GROUP
                                       CODE
                     71
                                       DATA
                           DGROUP GROUP
                      72
                                 ASSUME CS:CGROUP, DS:DGROUP, ES:DGROUP
                      73
                      74
                                 PUBLIC RFUN3L
                      75
                           76
                      77
                      78
                                 STACK STRUCTURE
                      79
                      80
                           PARAMS STRUC
                      81
                      82
                           OLD_ES DW
                                             ;SAVED ES REGISTER
0000
                           OLD_BP DW
                                       ?
                                             :SAVED BP REGISTER
                      83
0002
                                       ?
                                             ;SAVED DS REGISTER
                      84
                           OLD_DS DH
 0004
                                             RETURN ADDRESS (OFFSET ON TOP, THEN SEGMENT)
                                       ?
                      85
                           RETURN DD
0006
                                             ; ADDRESS OF DATA STRUCTURE (OFFSET, THEN SEGMENT)
                           DADDR DD
                                       ?
 000A
                      86
                                             ; ADDRESS OF ZV (OFFSET, THEN SEGMENT)
                      87
                                 DD
                                       ?
 000E
                           Z٧
                                             ; ADDRESS OF XV (OFFSET, THEN SEGMENT)
                                 DD
 0012
                      88
                           XΥ
                      89
                      90
                           PARAMS ENDS
                      91
                           92
                           93
                      94
                      95
                                 SEGMENT PUBLIC 'DATA'
                           DATA
                      96
                      97
                           STUSAVE DW
                                       ?
 0000 ????
                      98
                      99
                           DATA
                                 EMDS
                     100
                           101
                            102
                     103
                                SEGMENT PUBLIC 'CODE'
                     104
                           CODE
                     105
```

LOC OBJ		LINE	SOURCE	•			
0000		106	FARCOD	E PROC	FAR		
		107					
		108					
		109					
0000		110	RFUNGL				
0000 1E 0001 55		111		PUSH	DS	# SAVE	REGISTERS
0001 33		112		PUSH	BP FC		
0002 08 0003 8BEC		113 114		PUSH MOV	ES BP, SP		
0005 B8	R	115		VOM	AX, DATA		
0009 8EC0		116		YON	ES, AX		
000A C55E12		117		LDS	BX, [BP].XV	:GFT	XV ADDRESS AND SEGMENT
000D 9BD907		118		FLD	DWORD PTR [BX]		XV ON TOP OF 8087 STACK
0010 C55E0E		119		LDS	BX, [BP].ZV		ZV ADDRESS AND SEGMENT
0013 9BD907		120		FLD	DWORD PTR (BX)		ZV ON TOP OF 8087 STACK
0016 C55E0A		121		LDS	BX, [BP].DADDR	;GET	START ADDRESS OF DATA STRUCTURE
0019 8BD3		122		VOM	DX, BX	; SAVE	
001B 8B2F 001D 8B4F04		123		VOM	BP, [BX].XPTR		XPTR
0020 8B7F02		124		MOA	CX, [BX].NXPTS		NXPTS
0023 D1E7		125 126		VON	DI, [BX].ZPTR	;D1 =	· ZPTR
0025 D1E7		127		SHL SHL	DI, 1 DI, 1	• D7 -	DUTE INDEX - ELEMENT INDEX - A
0027 887706		128		VOH	SI, [BX].NZPTS		: BYTE INDEX = ELEMENT INDEX * 4 : NZPTS
002A D1E6		129		SHL	SI, 1	701 -	. 1127 13
002C D1E6		130		SHL	SI, 1	:SI =	: NO. OF BYTES = NZPTS * 4
		131				,	THE STATES HELICALLY
002E		132	GET_Z_	INDEX:			
002E 83C308		133		ADD	BX, 8	;BX =	START OF Z ARRAY
0031 9BD811	_	134 +2		FCOM	DWORD PTR (BX + DI)	;	COMPARE ARRAY ELEMENT WITH ZV
0034 9B26DD3E0000 003A 9B	R	135 +1		FSTSW	ES:STHSAVE		
003B 26A10000	R	135 +1 137 +1		FWAIT MOV	AV FOLOTUOAUS		
003F 250041	n	138 +1		AND	AX, ES:STWSAVE AX, 4100H		MACK IN COMPARISON DITO
0042 3D0001		139 +2		CMP	AX, 0100H	i	MASK-IN COMPARISON BITS
0045 7434		140 +2		JE	LEFT_OF_Z_START		; EXIT IF ZV < ARRAY ELE
			HENT				/ EATH IN 24 C HARMY ELE
		141 +2					
		142				;IF Z	> DATA(ORIGINAL INDEX)
0047		143	RIGHT_	OF_Z_STAP		; ***	THEN BEGIN ***
0047 830704		144		ADD	DI, 4		NDEX = INDEX + 4
004A 3BFE		145	RZS1:		DI, SI		HILE (INDEX <= NG. GF BYTES)
004C 741B		146 147		JE	RZS2	;	non EO EEGEN
004E 9BD811		148 +2		FCOM	רות + עמן פדם תפהשת	•	*** DO BEGIN *** COMPARE ARRAY ELEMENT WITH ZV
0051 9B26DD3E0000	R	149 +1		FSTSW	ES: STWSAVE	,	COMPANE HANN ELEMENT WITH IV
0057 9B		150 +1		FWAIT	EC-CIMONIE		
0058 25A10000	R	151 +1		YOH	AX, ES:STHSAVE		
0050 250041		152 +1		AND	AX, 4100H	;	MASK-IN COMPARISON BITS
005F 3D0001		153 +2		CHP	AX, 0100H		
0062 7405		154 +2		JΕ	RIS2	;	EXIT IF ZV < ARRAY ELEMENT
0064 830704		155 +2		ADD	T) T 4		<b></b>
0067 EBE1		156 157		add Jmp	DI, 4 RZS1	;	INDEX = INDEX + 4
0069 8BC6		159	RZS2:	MOV	AX, SI	;	*** END (WHILE) ***
006B 00		159		SUB	AX, 4	, ,	
					*****	•	

LOC OBJ	LINE	SOURCE		
006E 83EF04	160	SU		i
0071 3BF8	161	CM		; IF INDEX $\bigcirc$ NO. OF BYTES
0073 7542	162	JN		; THEN FOUND_LOW_INDEX
0075 9BDDD8	163	FS		; ELSE OUT_CF_RANGE
0078 EB4D90	164	JH	P GET_X_INDEX	; 
	165			;*** END (IF) ***
	166			; IF Z < DATA(ORIGINAL INDEX)
007B	167	LEFT_OF_Z_		;*** THEN BEGIN ***
007B 83EF04	168	SU		; INDEX = INDEX - 4
007E 83FF00	169	LZS1: CF		; WHILE (INDEX > 0)
0081 7020	170	J.	LZS2	· · · · · · · · · · · · · · · · · · ·
	171			; *** DO BEGIN ***
0083 9BD811	172 +2		OM DWORD PTR (BX + DI)	; COMPARE ARRAY ELEMENT WITH ZV
** *	R 173 +1		TSW ES:STWSAVE	
008C 9B	174 +1		AIT	
	R 175 +1			. MACK IN CONDADICON DITC
0091 250041	176 +1			; MASK-IN COMPARISON BITS
0094 3D0000	177 +2			; EXIT IF ZV > ARRAY ELEMENT
0097 740A	178 +2			
0099 3B0040	179 +2			; EXIT IF ZV = ARRAY ELEMENT
0090 7405	180 +2			
009E 83EF04	181	SU		
OOA1 EBDB	182	JM		
00A3 83FF00	183	LZS2: CM		; IF INDEX (> 0
00A6 7B0F	184 185	JG Ar		; THEN FOUND_LOW_INDEX ; ELSE OUT_OF_RANGE
00A8 83C704 00AB 9BDDD3	186	AI F9		: ELSE OUT_OF_RANGE
	187	FL FL		; REPLACE TOP WITH 0;
00AE 9BD9EE 00B1 9BD9C9	188	_	DZ CH ST(1)	; REFLACE for With V:
00B4 EB1190	189			7
0004 601170	190	Ur	P GET_A_INDEA	, ;*** END(IF) ***
	191			YAR CUDITY ARK
00B7	192	FOUND_LOW_	7 INDEX:	
00B7 9BD901	193	FL		STORE ZL ON STACK TOP
OOBA 9BDCE9	194		UB ST(1), ST	;(ZV-ZL) 2ND FROM TOP
00BD 9BD86904	195		UBR DWORD PTR [BX+DI+4]	
OOC1 9BDEF9	196	FI	IV	;(ZV-ZL)/(ZH-ZL) LEFT ON STACK
00C4 9BD9C9	197	F	CH ST(1)	;PUT XV ON TOP OF 8087 STACK
	198			
00C7	199	GET_X_INDE	<b>X:</b>	
00C7 03DE	200	AI	D BX, SI	;BX = START OF XARRAY
00C9 8BC7	201	MO	V AX, DI	:AX = NO. OF Z BYTES
00CB F6E1	202	ML	L CL	AX = XC ROW OFFSET=N. OF ZBYTES*NXPTS
00CB 03D8	293	AI	D BX, AX	;BX = XC ROW PTR
00CF 87FD	204	XC	HG DI.BP	<pre>;DI = XPTR : BP = LOW_Z_INDEX</pre>
00D1 D1E7	205	SI	L DI, 1	
00D3 D1E7	206	SH		;DI = BYTE INDEX = XPTR * 4
00D5 87F1	207		HG SI, CX	;SI = NXPTS : CX = NO. OF Z BYTES
00D7 D1E6	208	SH		I NO OF A PATEO NAME A
00D9 D1E6	209	Si		SI = NO. OF X BYTES = NXPTS * 4
00DB 9BD901	210	FL		STORE XC ON TOP OF STACK
00DE 8BC5	211	M(		
00E0 050400	212	AI		
00E3 3BC1	213	C?		FRIT IS 7 TOO HIGH
00E5 7410	214	J5	GX1	EXIT IF Z TOO HIGH

LOC OBJ		LINE	SOURCE				
00E7 03DF		215		ADD	BX, DI		
00E9 9BD900		216		FLD	DHORD PTR [BX + SI]	STORE	E XB ON TOP OF STACK
OOEC 2BDF		217		SUB	BX, DI		
OOEE 9BD8E1		218		FSUB	ST, ST(1)		C ON ST. TOP
00F1 9BD8CB 00F4 9BDEC1		219 220		FAUL	ST, ST(3)		-ZL)/(ZH-ZL))(XB-XC) ON ST. TOP
00F7 9BD8D1		221 +2	GV1.	FADD FCOM	ST(1)		ST. TOP
00FA 9B26DD3E0000	R	222 +1	GAI.	FSTSW	ES:STWSAVE	;	COMPARE ARRAY ELEMENT WITH XV
0100 9B	••	223 +1		FWAIT	LOTOTRONYL		
0101 26A10000	R	224 +1		VOM	AX, ES:STWSAVE		
0105 250041		225 +1		AND	AX, 4100H	;	HASK-IN COMPARISON BITS
0108 3D0000		226 +2		CMP	AX, 0000H	•	EXIT IF XV > ARRAY ELEMENT
010B 745C		227 +2		JΕ	LEFT_OF_X_START		
		228 +2 229				. T	3 DATA (COTOTNAL TARREY)
0100		230	RIGHT (	OF_X_STA	RT:		> DATA(ORIGINAL INDEX) HEN BEGIN ***
010D 83C704		231	************	ADD	DI, 4		IDEX = INDEX + 4
0110 3BFE		232	RXS1:	CMP	DI, SI		HILE (INDEX <= NO. OF BYTES)
0112 743D		233		JΕ	RXS2	ţ	
0444 000004		234				;	*** DO BEGIN ***
0114 9BD901		235		FLD	DWORD PTR (BX + DI)	;	STORE XD ON TOP OF STACK
0117 8BC5 0119 050400		236 237		MOV ADD	AX, BP AX, 4	;	
011C 3BC1		238		CMP	AX, CX	;	
011E 7410		239		JE	RXS3	;	EXIT IF Z TOO HIGH
0120 03DF		240		ADD	BX, DI	;	EAT1 11 2 100 1110/1
0122 9BD900		241		FLD	DWORD PTR [BX + SI]	;	STORE XE ON TOP OF STACK
0125 2BDF		242		SUB	BX, DI	i	
0127 9BDSE1 012A 9BD8CC		243		FSUB	ST, ST(1)	i	XE-XD ON ST. TOP
012D 9BDEC1		244 245		FAUL Fadd	ST, ST(4)	;	((ZV-ZL)/(ZH-ZL))(XD-XE) ON ST. TOP
0130 9BD8D2		246 +2	RXS3:	FCOM	ST(2)	;	XG ON ST. TOP COMPARE ARRAY ELEMENT WITH XV
0133 9B26DD3E0000	R	247 +1		FSTSW	ES: STWSAVE	•	Court and transfer profit to the transfer of t
0139 98		248 +1		FWAIT			
013A 26A10000	R	249 +1		VOM	AX, ES:STHSAVE		
013E 250041		250 +1		AND	AX, 4100H	ī	MASK-IN COMPARISON BITS
0141 3D0000 0144 740B		251 +2 252 +2		CMP JE	AX, 0000H	ţ	EXIT IF XV > ARRAY ELEMENT
0111 7400		253 +2		UE.	RXS2		
0146 9BD9C9		254		FXCH	ST(1)	;	XF = XG
0149 9BDDD8		255		FSTP	ST(0)	;	DISCARD OLD XF
0140 830704		256		ADD	DI, 4	;	INDEX = INDEX + 4
014F EBBF		257	5466.	JMP	RXS1	;	*** END (WHILE) ***
0151 9BD9C9 0154 8BC6		258 259	RXS2:	FXCH	ST(1)	;	
0154 0D00		260		MOV SUB	AX, SI AX, 4	7	
0159 83EF04		261		SUB	DI, 4	;	
015C 3BF8		262		CHP	DI, AX	; IF	INDEX <> NO. OF BYTES
015E 7567		263		JNE	FOUND_LOW_X_INDEX	;	THEN FOUND_LOW_INDEX
0150 9BDDD8		264		FSTP	ST(0)	;	ELSE OUT_OF_RANGE
0163 9BDDD8 0166 EB7490		265		FSTP	ST(0)	;	DISCARD XG, AND XV
VIOO ED/47V		266 267		JMP	GET_ANSWER	; **** =	ND /IEL xxx
		268					ND (IF) *** C DATA(ORIGINAL INDEX)
0169		269	LEFT_OF	_X_START	7		HEN BEGIN ***
							·-·· - <del></del> ·

0169 86F04	LOC OBJ		LINE	SOURCE				
September   1	0140 000004		270		CITE	DT. A	•	INDEY = INDEY - 4
OLF TO				1 451:	-			
1				LAGI.				THE LANGE TO STATE OF THE STATE
1774   SECS	010F 7642					CAGE	•	*** DO BEGIN ***
0174 SECS  0176 O09400  0276 ABD AX, 4  0177 30C1  0277 CMP AX, 4  0177 30C1  0278 JE LIS3  1 EXIT IF Z TOO HIGH  1070 30C7  0279 ADD BX, 01  1077 90C0  0279 ADD BX, 01  1077 90C0  0279 ADD BX, 01  1077 90C0  0279 ADD BX, 01  1070 90C0  0279 ADD BX, 01  1070 90C0  0279 ADD BX, 01  1070 90C0  0279 ADD BX, 01  1 STORE XE ON TOP OF STACK  0182 2E0F 281 SUB BX, 01  1 STORE XE ON TOP OF STACK  0183 90C0C1  0283 FRIL ST, ST(1) STACH XE ON ST, TOP  0184 90C0C1  0284 FADD  0195 90C0C2  0285 +1 FADD  0196 90C0C2  0285 +1 FADD  0197 90C0C0C0C0C0  0286 +1 FSTSH SISTHSAWE  0198 250C41  0287 +1 AND AX, 4100H  0198 250C41  0287 +2 LS2  0198 250C41  0287 +2 LS2  0198 250C41  0287 +2 LS2  0198 250C41  0288 +1 AND AX, 4100H  0198 250C0C1  029 +2 CPP AX, 4000H  0198 250C0C0  029 42 CPP AX, 4000H  0198 250C0C0  0198 250C0C0  029 42 FXCH  0198 250C0C0  0199 250C0	0171 980901				FLD	DWORD PTR [BX + DI]	;	
0175   029400							;	
1775   2017   277							ţ	
177   1410   278							;	
1017   1020   280							;	EXIT IF Z TOO HIGH
SUB-	017D 03DF		279		ADD	BX, DI	;	
1018   2018   2018   2018   2019   2018   2019   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018   2018	017F 9BD900		280		FLD	DWORD PTR [BX + SI]	;	STORE XE ON TOP OF STACK
1987 98BBCC	0182 2BDF		281		SUB	BX, DI	7	
0188 9ETECT 294 FADD 57(2)	0184 9BD8E1		282				;	
OIRD 980802	0187 9BD8CC		283			ST, ST(4)	;	
O190 9826E0520000   R   286 +1	018A 9BDEC1						5	· -
0196 9B				LXS3:			;	COMPARE ARRAY ELEMENT WITH XV
0197 26A10000 R 288 +1 NOV AX, ES:STNSAVE 0198 259041 289 +1 AND AX, 4100H 0198 259041 290 +2 CMP AX, 4100H 01A1 7410 291 +2 JE LXS2 \$ EXIT IF XV < ARRAY ELEMENT 01A3 360040 292 +2 CMP AX, 4000H 01A6 740B 293 +2 JE LXS2 \$ EXIT IF XV < ARRAY ELEMENT 01A8 360040 292 +2 JE LXS2 \$ EXIT IF XV < ARRAY ELEMENT 01A8 98D0C9 294 FXCH ST(1) \$ XF=XG 01AB 98D0C9 295 FSTP ST(0) \$ IDISCARD OLD XF 01AB 38D0C9 295 FSTP ST(0) \$ IDISCARD OLD XF 01AB 38D0C9 297 JMP LXS1 \$ *** END (MHILE) *** 01B 383FF00 298 LXS2: CMP DI, 0 \$ IF INDEX < 0 01B 583FF00 298 LXS2: CMP DI, 0 \$ IF INDEX < 0 01B 70F 299 JMP EXXID \$ INDEX = INDEX A 01B 98D0CB 301 FSTP ST(0) \$ IDISCARD XC 01B 98D0CB 301 FSTP ST(0) \$ IDISCARD XC 01B 98D0CB 301 FSTP ST(0) \$ IDISCARD XC 01C 98D9EE 303 FLBZ \$ INDEX \$ INDEX EXPLOYED AX 01C 4EB1690 304 JMP GET_ANSWER 01C7 307 FOUND_LOAL_XINDEX: \$ XF, XG, XV, ZSLOPE ON ST. 01C7 98DCEA 308 FSUB ST(2), ST \$ XF, XG, XV, ZSLOPE ON ST. 01C7 98DCEA 308 FSUB ST(2), ST \$ XF, XG, XV, XF, XG, XV-FF), ZSLOPE 01C0 98DCF 310 FDUV \$ (XM-FF), ZSLOPE 01C1 98DFC 311 MOV AX, BP \$ IF Z NOT TOO HIGH 01D SBC5 311 MOV AX, BP \$ IF Z NOT TOO HIGH 01D SBC5 311 MOV AX, BP \$ IF Z NOT TOO HIGH 01D SBC5 311 MOV AX, BP \$ IF Z NOT TOO HIGH 01D SBC5 311 MOV AX, SP \$ IF Z NOT TOO HIGH 01D SBC5 311 MOV AX, SP \$ IF Z NOT TOO HIGH 01D SBC5 311 MOV AX, SP \$ IF Z NOT TOO HIGH 01D SBC5 311 MOV AX, SP \$ IF Z NOT TOO HIGH 01D SBC5 311 MOV AX, SP \$ IF Z NOT TOO HIGH 01D SBC5 \$ SIS MOV AX, SI \$ IN EN (ZV-ZL)/(ZH-ZL) ON ST, TOP 01D SBC5 \$ SIS MOV AX, SI \$ IN EN (ZV-ZL)/(ZH-ZL) ON ST, TOP 01D SBC6 \$ SIS MOV AX, SI \$ IN EN (ZV-ZL)/(ZH-ZL) ON ST, TOP 01D SBC6 \$ SIS MOV AX, SI \$ IN EN (ZV-ZL)/(ZH-ZL) ON ST, TOP 01D SBC6 \$ SIS MOV AX, SI \$ IN EN (ZV-ZL)/(ZH-ZL) ON ST, TOP 01D SBC6 \$ SIS MOV AX, SI \$ IN EN (ZV-ZL)/(ZH-ZL) ON ST, TOP 01D SBC7 \$ SIS MAX 1 \$ IN EN (ZV-ZL)/(ZH-ZL) ON ST, TOP 01D SBC7 \$ SIS MAX 1 \$ IN EN (ZV-ZL)/(ZH-ZL) ON ST, TOP 01D SBC8 \$ SIS MAX 1 \$ IN EN (ZV-ZL)/(ZH-ZL) ON ST, TOP 01D SBC9 \$ SIS MAX 1 \$ IN EN (ZV-ZL)/(ZH-ZL) ON ST, TOP 01D SBC9 \$ SIS MAX 1 \$ IN		R				ES:STWSAVE		
019B 250041 289 +1 AND AX, 4100H 5 MASK-IN COMPARISON BITS 019E 300001 290 +2 CMP AX, 0100H 0141 7410 291 +2 JE LXS2 5 EXIT IF XV < ARRAY ELEMENT 01A3 300040 292 +2 CMP AX, 4000H 01A6 740B 293 +2 JE LXS2 5 EXIT IF XV < ARRAY ELEMENT 01A8 9BDC9 294 FKCM ST(1) 5 XF=XG 01AB 9BDD08 295 FSTP ST(0) 5 DISCARD OLD XF 01AE 33EF04 296 SUB DI, 4 5 INDEX = INDEX = 4 01B1 EBB9 297 JPE KS1 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1								
19FE 300001		R						MACK TH COMPADICON DITC
OTAT   7410   291 +2							;	MHOK-IN COMPHATOON BITO
0143 3D0040 272 +2 CMP AX 4000H 0143 740B 273 +2 JE LXS2 ; EXIT IF XV = ARRAY ELEMENT 0143 7809C9 294 FXCH ST(1) ; XF=XG 0148 7809C9 294 FXCH ST(1) ; XF=XG 0148 9809C9 295 FSTP ST(0) ; DISCARD 0.D XF 0148 9809C8 295 FSTP ST(0) ; DISCARD 0.D XF 0148 83EF04 296 SUB DI, 4 ; INDEX = INDEX = A 0181 83FF00 298 LXS2: CMP DI, 0 ; IF INDEX < 0 0183 83FF00 298 LXS2: CMP DI, 0 ; IF INDEX < 0 0184 780F0 299 JGE FOUND_LON_X_INDEX ; THEN FOUND_LON_INDEX 0185 83C704 300 ADD DI, 4 ; ELSE OUT_OF_RANGE 0186 980D03 301 FSTP ST(0) ; DISCARD XC 0187 980D03 302 FSTP ST(0) ; DISCARD XC 0106 980EB 300 FSTP ST(0) ; DISCARD XC 0107 980FE 303 FLDZ ; REPLACE TOP HITH 0 0107 980FE 303 FSUB ST(2), ST								EVIT TE VIL A ADDAY ELEMENT
OTA			_				,	EXII IL YA C HUUHI EFELEDI
0188 9BD0799							•	EYIT IE YU = ARRAY FLEMENT
01AB 9BDDB8								
01AE 83EF04 296 SUB DI, 4 ; INDEX = INDEX − 4 01B1 EBB9 297 JAP LXS1 ; *** END (HHILE) *** 01B3 83FF00 298 LXS2: CMP DI, 0 ; IF INDEX < 0 01B6 7B0F 299 JGE FOUND_LON_X_INDEX ; THEN FOUND_LON_LINDEX 01B8 83CF04 300 ADD DI, 4 ; ELSE OUT_OF_RANGE 01BB 9BDDD8 301 FSTP ST(0) ; 01BE 9BDDD8 302 FSTP ST(0) ; DISCARD XC 01C1 9BD9EE 303 FLDZ ; REPLACE TOP HITH 0 01C4 EB1690 304 JMP GET_ANSWER ; 01C7 9BDCEA 308 FSUB ST(2), ST (XF, XG, XV, ZSLOPE ON ST. 01C7 9BDCEA 309 FSUB (XG-XF), (XV-XF), ZSLOPE O1C0 9BDEF9 310 FDIV (XG-XF), XF-XF), ZSLOPE 01C0 9BDEF9 310 FDIV (XG-XF), XF-XF), ZSLOPE 01C0 9BDEF9 310 FDIV (XG-XF), ZSLOPE 01C0 9BDEF9 310 FDIV (XG-XF), ZSLOPE 01C0 SBC1 313 CMP AX, 4 01C5 SBC1 313 CMP AX, 4 01C5 SBC1 313 CMP AX, 4 01C5 SBC1 313 CMP AX, CX 01C9 7403 314 JE GET_ANSWER 01C9 9BD9C9 315 FXCH ST(1) ; THEN (ZV-ZL)/(ZH-ZL) ON ST. TOP 316 01C0 BBC6 316 MOV AX, SI ; AX = NO. OF X BYTES 01CE DIEB 319 SHR AX, 1 ; AX = NO. OF X BYTES 01CE NO FZ EYTES 14X = NO. OF X BYTES 14X = NO. OF X BYT								• • • • • • • • • • • • • • • • • • • •
01B1 EBB9								
0183 83FF00							;	
O126 7DOF				LXS2:			;	
01B8 83C704 300 ADD DI, 4 ; ELSE OUT_OF_RANGE 01BB 9BDD8 301 FSTP ST(0) ; 01EC 9BDD08 302 FSTP ST(0) ; DISCARD XC 01C1 9BD9EE 303 FLDZ ; REPLACE TOP WITH 0 01C4 EB1690 304 JMP GET_ANSWER ; 305							;	THEN FOUND_LOW_INDEX
OIBB 9BDDB8   301						DI, 4	;	ELSE OUT_OF_RANGE
OIBE   9BDDB3   302					FSTP	ST(0)	;	
01C4 EB1690  304  305  306  01C7  307  5000 FOUND_LOH_X_INDEX:  1XF, XG, XV, ZSLOPE ON ST.  1XF, XG, XV-XF), ZSLOPE  1XF, XG, (XV-XF), ZSLOPE  1XF, XG, (XV-XF), ZSLOPE  1XF, XG, (XV-XF), ZSLOPE  1XF, XG, (XV-XF), ZSLOPE  1XG-XF), XG-XF, XG, XG  1XG-XF, XG, XG  1XG-XF, XG, XG  1XG-XF, X			302		FSTP	ST(0)	;	
305   306   307	01C1 9BD9EE		303		FLDZ		;	REPLACE TOP WITH O
306   307   FOUND_LOH_X_INDEX:   XF, XG, XV, ZSLOPE ON ST.	01C4 EB1690		304		JMP	GET_ANSWER	•	
01C7 9DCEA 308 FSUB ST(2), ST (XF, XG, XV, ZSLOPE ON ST. 01C7 9DDCEA 308 FSUB ST(2), ST (XF, XG, (XV-XF), ZSLOPE 01CA 9DDEE9 309 FSUB ((XV-XF), (XV-XF), ZSLOPE 01CD 9DDEF9 310 FDIV ((XV-XF)/(XG-XF)), ZSLOPE 01D0 8BC5 311 MOV AX, BP ((XV-XF)/(XG-XF)), ZSLOPE 01D2 050400 312 ADD AX, 4 01D5 3BC1 313 CMP AX, CX 01D7 7403 314 JE GET_ANSWER 01D9 9BD9C9 315 FXCH ST(1) THEN (ZV-ZL)/(ZH-ZL) ON ST. TOP 316 01DC 317 GET_ANSWER: 01DC 8BC6 318 MOV AX, SI (AX = NO. OF X BYTES) 01DE DIE8 319 SHR AX, 1 01ED DIE8 320 SHR AX, 1 01E0 DIE8 320 SHR AX, 1 01E2 F6E1 322 MUL CL (AX = NXPTS = NO. OF X BYTES) 01E2 F6E1 322 MUL CL (AX = XV =							5 <del>* * 1</del>	* END(IF) ***
01C7 9PDCEA 308 FSUB ST(2), ST ;XF, XG, (XV-XF), ZSLOPE 01CA 9PDEE9 309 FSUB ;(XG-XF), (XV-XF), ZSLOPE 01CD 9PDEF9 310 FDIV ;((XV-XF)/(XG-XF)), ZSLOPE 01D0 8BC5 311 MOV AX, BP ;IF Z NOT TOO HIGH 01D2 050400 312 ADD AX, 4 01D5 3BC1 313 CMP AX, CX 01D7 7403 314 JE GET_ANSWER 01D9 9BD9C9 315 FXCH ST(1) ; THEN (ZV-ZL)/(ZH-ZL) ON ST. TOP 316 01DC 317 GET_ANSWER: 01DC 317 GET_ANSWER: 01DC 317 GET_ANSWER: 01DC 317 GET_ANSWER: 01DC 318 MOV AX, SI ;AX = ND. OF X BYTES 01DE D1E8 319 SHR AX, 1 01ED D1EB 320 SHR AX, 1 01EO D1EB 320 SHR AX, 1 01E2 F6E1 322 MUL CL ;AX = YC RON OFFSET 323 ;THE AVOVE INSTR. LIMITS LENGTH OF ZARRAY						mev.		VO VII TOLOGE ON CT
01CA 9BDEE9 309 FSUB ;(XG-XF), (XV-XF), ZSLOPE 01CD 9BDEF9 310 FDIV ;((XV-XF)/(XG-XF)), ZSLOPE 01D0 8BC5 311 MOV AX, BP ;IF Z NOT TOO HIGH 01D2 050400 312 ADD AX, 4 01D5 3BC1 313 CMP AX, CX 01D7 7403 314 JE GET_ANSWER 01D9 9BD9C9 315 FXCH ST(1) ; THEN (ZV-ZL)/(ZH-ZL) ON ST. TOP 316 01DC 317 GET_ANSWER: 01DC 8BC6 318 MOV AX, SI ;AX = NO. OF X BYTES 01DE DIE8 319 SHR AX, 1 01E0 DIEB 320 SHR AX, 1 01E0 DIEB 321 ;CX = NO OF Z BYTES 01E2 F6E1 322 MUL CL ;AX = YC ROH OFFSET 323 ;THE AVOVE INSTR. LIMITS LENGTH OF ZARRAY				FUUNU_				
01CD 9EDEF9 310 FDIV ;((XV-XF)/(XG-XF)), ZSLOPE 01D0 8BC5 311 MOV AX, BP ;IF Z NOT TOO HIGH 01D2 050400 312 ADD AX, 4 01D5 3BC1 313 CMP AX, CX 01D7 7403 314 JE GET_ANSWER 01D9 9BD9C9 315 FXCH ST(1) ; THEN (ZV-ZL)/(ZH-ZL) ON ST. TOP 316 01DC 317 GET_ANSWER: 01DC 8BC6 318 MOV AX, SI ;AX = NO. OF X BYTES 01DE DIEB 319 SHR AX, 1 01ED DIEB 320 SHR AX, 1 01ED DIEB 321 ;CX = NO OF Z BYTES 01E2 F6E1 322 MUL CL ;AX = YC ROW OFFSET 323 ;THE AVOVE INSTR. LIMITS LENGTH OF ZARRAY						51121, 51		
01D0 8BC5								
01D2 050400 312 ADD AX, 4 01D5 3BC1 313 CMP AX, CX 01D7 7403 314 JE GET_ANSWER 01D9 9BD9C9 315 FXCH ST(1) ; THEN (ZV-ZL)/(ZH-ZL) ON ST. TOP 316 01DC 317 GET_ANSWER: 01DC 8BC6 318 MOV AX, SI ;AX = NO. OF X BYTES 01DE DIE8 319 SHR AX, 1 01E0 DIE8 320 SHR AX, 1 01E0 DIE8 321 ;CX = NO OF Z BYTES 01E2 F6E1 322 MUL CL ;AX = YC ROW OFFSET 323 ;THE AVOVE INSTR. LIMITS LENGTH OF ZARRAY						ΔY. RP		
01D5 3BC1 313 CMP AX, CX 01D7 7403 314 JE GET_ANSWER 01D9 9BD9C9 315 FXCH ST(1) ; THEN (ZV-ZL)/(ZH-ZL) ON ST. TOP 316 01DC 317 GET_ANSWER: 01DC 8BC6 318 MOV AX, SI ;AX = NO. OF X BYTES 01DE DIE8 319 SHR AX, 1 01E0 DIE8 320 SHR AX, 1 01E0 DIE8 321 ;CX = NO OF Z BYTES 01E2 F6E1 322 MUL CL ;AX = YC ROW OFFSET 323 ;THE AVOVE INSTR. LIMITS LENGTH OF ZARRAY							7 21	2 101 100 11251
01D7 7403 314 JE GET_ANSWER 01D9 9BD9C9 315 FXCH ST(1) ; THEN (ZV-ZL)/(ZH-ZL) ON ST. TOP 316 01DC 317 GET_ANSWER: 01DC 8BC6 318 MOV AX, SI ;AX = NO. OF X BYTES 01DE DIE8 319 SHR AX, 1 01E0 DIE8 320 SHR AX, 1 01E0 DIE8 321 ;CX = NO OF Z BYTES 01E2 F6E1 322 MUL CL ;AX = YC ROW OFFSET 323 ;THE AVOVE INSTR. LIMITS LENGTH OF ZARRAY								•
01D9 9BD9C9 315 FXCH ST(1) ; THEN (ZV-ZL)/(ZH-ZL) ON ST. TOP  01DC 316  01DC 317 GET_ANSWER:  01DC 8BC6 318 MOV AX, SI ;AX = NO. OF X BYTES  01DE D1E8 319 SHR AX, 1  01E0 D1E8 320 SHR AX, 1  01E0 F6E1 322 MUL CL ;AX = NO OF Z BYTES  01E2 F6E1 323 FXCH AV, 1  1								
316     317   GET_ANSWER:							;	THEN (ZV-ZL)/(ZH-ZL) ON ST. TOP
01DC 317 GET_ANSWER: 01DC 8BC6 318 MOV AX, SI 1AX = NO. OF X BYTES 01DE D1E8 319 SHR AX, 1 01E0 D1E8 320 SHR AX, 1 321 ;CX = NO OF Z BYTES 01E2 F6E1 322 MUL CL 1AX = YC ROW OFFSET 323 ;THE AVOVE INSTR. LIMITS LENGTH OF ZARRAY	0107 700707					•		
01DC 8BC6       318       MOV       AX, SI       ;AX = NO. OF X BYTES         01DE D1E8       319       SHR       AX, 1       ;AX = NXPTS = NO. OF X BYTES / 4         01E0 D1E8       320       SHR       AX, 1       ;AX = NXPTS = NO. OF X BYTES / 4         01E2 F6E1       322       MUL       CL       ;AX = YC ROH OFFSET         323       ;THE AVOVE INSTR. LIMITS LENGTH OF ZARRAY	01DC			GET_AN	ISWER:			
01PE D1E8 319 SHR AX, 1 01E0 D1E8 320 SHR AX, 1 321 ;CX = NO OF Z BYTES / 4 321 ;CX = NO OF Z BYTES 01E2 F6E1 322 MUL CL ;AX = YC ROH OFFSET 323 ;THE AVOVE INSTR. LIMITS LENGTH OF ZARRAY						AX, SI	;AX	= NO. OF X BYTES
321 ; CX = NO OF Z BYTES 01E2 F6E1 322 MUL CL ; AX = YC ROH OFFSET 323 ; THE AVOVE INSTR. LIMITS LENGTH OF ZARRAY					SHR			
01E2 F6E1 322 MUL CL SAX = YC ROH OFFSET 323 STHE AVOVE INSTR. LIMITS LENGTH OF ZARRAY	01E0 D1E8		320		SHR	AX, 1		
323 ; THE AVOVE INSTR. LIMITS LENGTH OF ZARRAY								
VAN	01E2 F6E1				MLL	CL		
324 \$ 10 255 ELEMENTS								
			324			•	;10	ZOO ELEMENIS

£00 0	DBJ	LINE	SOURCE			
01E4 0	)3D8	325		ADD	BX, AX	BX = YC ROW PTR = NXPTS * ZPTR
01E6 (	)3DF	326		ADD	BX, DI	BX = YC POINTER
01E8 9	PBD907	327		FLD	DWORD PTR [BX]	STORE YC ON ST TOP
01EB 8	BBC5	328		HOV	AX, EP	
01ED 0	)50400	329		ADD	AX, 4	
01F0 3	BBC1	330		CHP	AX, CX	
01F2 7	7510	331		JNE	Z_NOT_HIGH	
		332				
01F4		333	Z_T00_H1	IGH:		; IF Z OUT OF RANGE
01F4 8	BBC7	334		MOV	AX, DI	:*** DO BEGIN ***
01F6 0	)50400	335		ADD	AX, 4	; IF X OUT OF RANGE
01F9 3	3BC6	336		CMP	AX, SI	
01FB 7	7444	337		JΕ	FOUND_YV	; THEN EXIT
01FD		338	X_VALID	Ī		; (Z TOO HIGH BUT X IS IN RANGE)
	PBD94704	339		FLD	DWORD PTR [BX + 4]	; STORE XD
0201 E	B3290	340		J₩ <del>P</del>	GET_YV	;*** END (IF) ***
		341				
0204		342	Z_NOT_H	(GH:		; IF Z VALID
0204 8		343		MOV	AX, DI	:*** DO BEGIN ***
0206 0		344		ADD	AX, 4	; IF X OUT OF RANGE
0209 3		345		CMP	AX, SI	1
020B 7	423	346		JE	Z_VALID	; THEN EXIT
020D	renno.	347	BOTH_VAL			; ELSE (BOTH XV AND ZV ARE IN RANGE)
020D 9		348		FXCH	ST(1)	;(ZV-ZL)/(ZH-ZL) ON TOP, THEN YC
0210 9		349		FLD	DWORD PTR [BX + SI]	STORE YB ON ST. TOP
0213 9		350		FSUB	ST, ST(2)	; (YB-YC) ON ST. TOP
0216 9		351 350		FMUL	ST, ST(1)	(((ZV-ZL)/(ZH-ZL))(YB-YC) ON ST TOP
0219 9	/BD94704	352		FADDP	ST(2), ST	TYF 2ND FROM TOP OF STACK
	BD94004	353		FLD	DWORD PTR [BX + 4]	STORE YD ON STACK TOP
0224 9		354 355		FLD	DWORD PTR [BX + SI + 4]	
0227 9		355 356		FSUB FMULP	ST, ST(1)	YE-YD ON ST TOP
022A 9		357		FADD	ST(2), ST	;((ZV-ZL)/(ZH-ZL))(YE-YD) 2ND FROM TOP
022D E		358		JMP	GET_YV	SYG ON TOP OF 8087
V	20070	359		Orn	OEILIV	;*** END (IF) ***
0230		360	Z_VALID:			; (X OUT OF RANGE BUT Z VALID)
0230 0	3DE	361		ADD	BX, SI	ATY OUT OF WHADE BOT I AHETBY
0232 9		362		FLD	DWORD PTR [BX]	
		363			Enough I III EDV3	
0235		364	GET_YV:			YG TOP, THEN YF, THEN SLOPE
0235 9	BD9E1	365		FSUB	ST, ST(1)	;YG-YF ON ST. TOP
0238 9		366		FMULP		;((XV-XF)/(XG-XF))(YG-YF) 2ND FROM TOP
023B 9	BDEC1	367		FADD		SYV ON ST. TOP
023E E	B0190	368		JHP	FOUND_YV	THE SHEET TO
		369				
		370				
0241		371	FOUNDLYV			
0241		372	SAVELNEW	_POINTE		
0241 8		373		VOM		RESTORE BX TO START OF DATA STRUC.
0243 D		374		SHR	DI, i	
0245 D:		375		SHR	DI, 1	
0247 8t		376		MOV		SAVE XPTR
0249 D:		377		SHR	BP, 1	
024B D		378		SHR	EP, 1	
024D 89	70FUZ	379		VOM	[BX].ZPTR, EP	;SAVE ZPTR

8086/87/88/186 MACRO ASSEM	IBLER	RFUN3L					08/30/84	PAGE	8
LOC OBJ	LINE	SOURCE							
	380								
0250	381	EPILOGU	<b>:</b> :						
0250 07	382		POP	ES					
0251 5D	383		P0P	₽P	RESTORE RE	GISTERS			
0252 1F	384		POP	DS					
0253 CA0C00	385		RET	12					
	386								
	387	FARCODE	ENDP						
	388								
	389	CODE	ENDS						
	390								
	391				<del>******************</del>				
	392	<del>}</del>							
	393								
	394		END						

ASSEMBLY COMPLETE, NO ERRORS FOUND

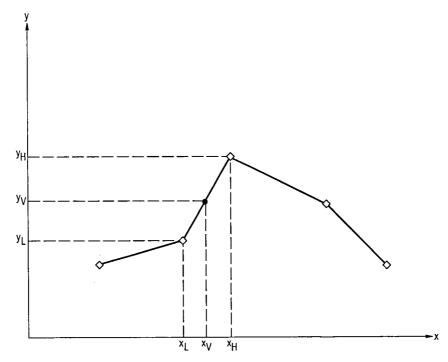


Figure 1. - Univariate function.

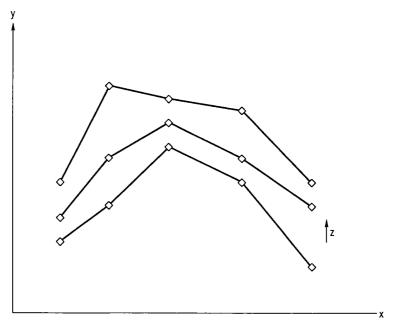


Figure 2. - Bivariate function.

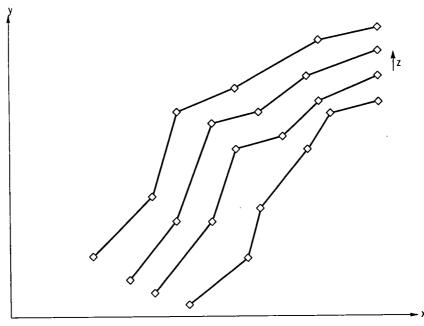


Figure 3. - Map function.

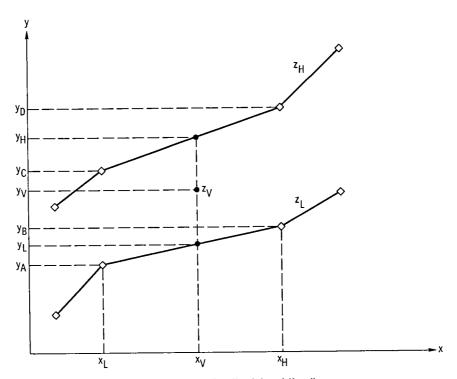


Figure 4. - Bivariate function interpolation diagram.

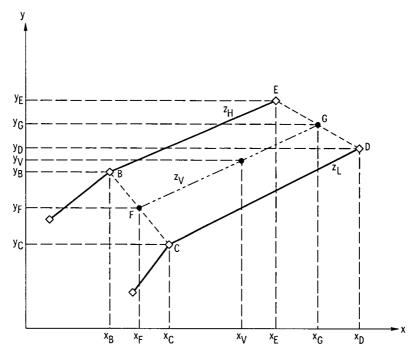


Figure 5. - Map function interpolation diagram.

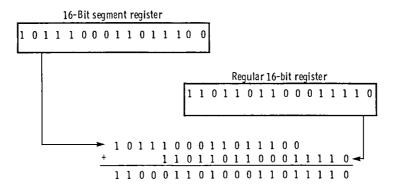


Figure 6. - How 16-bit registers are combined to form 20-bit address.

FIELD DEFINITION

XPTR: X pointer

ZPTR: Z pointer

NXPTS: Number of x data points per z curve

NZPTS: Number of z curves

Data Arrays: These fields hold actual experimental data points.

The points mark the start and end of a curve

Figure 7. - Field identifications in breakpoint information block.

DIMENSION XARRAY (7), YARRAY (7)
INTEGER XPTR, YPTR
COMMON / STUFF / XPTR, ZPTR, NXPTS, NZPTS, XARRAY, YARRAY
DATA XPTR, ZPTR, NXPTS, NZPTS / 0, 0, 7, 7 /
DATA XARRAY / 0, 0, 1, 0, 2, 0, 3, 0, 4, 0, 5, 0, 6, 0 /
DATA YARRAY / 0, 0, 1, 0, 4, 0, 9, 0, 16, 0, 25, 0, 36, 0 /
XIN = 1, 5
Y = RFUNIL (XIN, XPTR)
PRINT *, Y
END

Figure 8. - Fortran program for univariate function call.

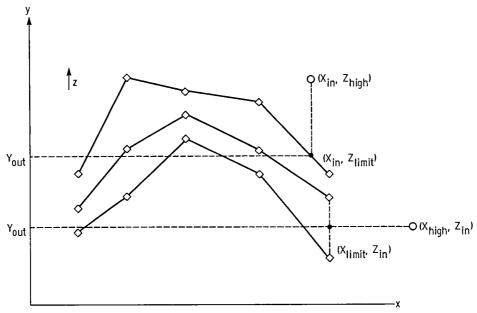


Figure 9. - Two out-of-range examples.

FIELD NAME	FIELD VALUE
XPTR	X index
ZPTR	Not used
NXPTS	Number of x data points (4 in this example)
NZPTS	Not used
XARRAY	X1, X2, X3, X4
YARRAY	Y1, Y2, Y3, Y4

Figure 10. - Breakpoint information block for RFUN1 programs.

FIELD NAME	FIELD VALUE
XPTR	X index
ZPTR	Z index
NXPTS	Number of x data points ( 4 in this example)
NZPTS	Number of z data points (5 in this example)
ZARRAY	Z1, Z2, Z3, Z4, Z5
XARRAY	X1, X2, X3, X4
YARRAY	Y11, Y12, Y13, Y14 Y21, Y22, Y23, Y24 Y31, Y32, Y33, Y34 Y41, Y42, Y43, Y44 Y51, Y52, Y53, Y54

Figure 11. - Breakpoint information block for RFUN2 programs.

FIELD NAME	FIELD VALUE
XPTR	X index
ZPTR	Z index
NXPTS	Number of x data points (4 in this example)
NZPTS	Number of z data points (5 in this example)
ZARRAY	Z1, Z2, Z3, Z4, Z5
XARRAY	X11, X12, X13, X14 X21, X22, X23, X24 X31, X32, X33, X34 X41, X42, X43, X44 X51, X52, X53, X54
YARRAY	Y11, Y12, Y13, Y14 Y21, Y22, Y23, Y24 Y31, Y32, Y33, Y34 Y41, Y42, Y43, Y44 Y51, Y52, Y53, Y54

Figure 12. - Breakpoint information block for RFUN3 programs.

				1 <b>0</b>
. Report No.	2. Government Accession N	io.	3. Recipient's Catalog N	
NASA TM-83783				
Title and Subtitle	<del>-</del>		5. Report Date	
			October 1984	* * *
Floating-Point Function G	ieneration Routines	s for	6. Performing Organizat	
16-Bit Microcomputers	•		505-43-4	
		·	8. Performing Organizat	ion Report No
7. Author(s)			6. Periorning Organizat	ion report No.
	. F. C. 1		E-2279	
Michael A. Mackin and Jar	nes F. Soeder		10. Work Unit No.	<del></del>
). Performing Organization Name and Address			11. Contract or Grant No	<u>.                                    </u>
National Aeronautics and	Space Administrat	ion	11. Contract of Grant No	•
Lewis Research Center				
Cleveland, Ohio 44135			13. Type of Report and P	eriod Covered
2. Sponsoring Agency Name and Address				
National Aeronautics and	Snace Administrat	ion	Technical Me	
Washington, D.C. 20546	Space Auministrat		14. Sponsoring Agency (	Jode .
Rushington, D.O. 20070				
			<u> </u>	<u> </u>
5. Supplementary Notes	•			
••				•
. *				
	• •			
		•		
of nonanalytic functions data in floating-point for 16-bit Intel 8086 system as fast as routines using routines are written in a modular fashion so as to	orm. However, bec with an 8087 math g data in scaled i assembly language,	ause they are ematical cope nteger form. they have be	e written for u rocessor, they Although all	use on a execute
	racilitate their	use with high		in a
	racilitate their	use with high		in a
	racilitate their	use with nig		in a
	racilitate their	use with nig		in a
	racificate their	use with nig		in a
	Tacilitate their	use with nig		in a
	Tacilitate their	use with nig		in a
	Tacilitate their	use with nig		in a
	Tacilitate their	use with nig		in a
	Tacilitate their	use with nig		in a
	Tacilitate their	use with nig		in a
	Tacilitate their	use with nig		in a
7 Kov Words (Suggested by Author(s))		•	h-level langua	in a
		l8. Distribution Statem	h-level languag	in a
Function generation		18. Distribution Statem Unclassifi	h-level languag nent ed - unlimited	in a
Function generation Look-up tables		l8. Distribution Statem	h-level languag nent ed - unlimited	in a
Function generation Look-up tables Floating-point routines		18. Distribution Statem Unclassifi	h-level languag nent ed - unlimited	in a
Function generation Look-up tables Floating-point routines Microprocesser control	1	8. Distribution Statem Unclassifi STAR Catego	h-level languag ent ed - unlimited ory 61	in a
Look-up tables Floating-point routines		8. Distribution Statem Unclassifi STAR Catego	h-level languag nent ed - unlimited	in a

National Aeronautics and Space Administration

Washington, D.C. 20546

Official Business
Penalty for Private Use, \$300

SPECIAL FOURTH CLASS MAIL POOK

LANGLEY HESEARCH CENTER

3 1176 00512 5886

| || || || || ||



Postage and Fees Paid National Aeronautics and Space Administration NASA-451

NASA

POSTMASTER:

If Undeliverable (Section 158 Postal Manual) Do Not Return